NLSY97 CODEBOOK SUPPLEMENT

MAIN FILE ROUND 4

Prepared for the
U.S. Department of Labor by

Center for Human Resource Research
The Ohio State University

Under contract with
National Opinion Research Center
University of Chicago

2002

This publication is prepared in conjunction with contract #J-9-J-9-0007 with the Bureau of Labor Statistics, U.S. Department of Labor.

TABLE OF CONTENTS

Introduction to the CAPI Questionnaire and Codebook i

NLSY97 Attachment 1: 1990 Census Industrial & Occupational Classification Codes 1

Introduction to the Created Variable Appendices 23

NLSY97 Appendix 1: Education Variable Creation 27

NLSY97 Appendix 2: Employment Variable Creation 69

NLSY97 Appendix 3: Family Background and Formation Variable Creation 251

NLSY97 Appendix 4: Geographic Variable Creation 271

NLSY97 Appendix 5: Income and Assets Variable Creation 275

NLSY97 Appendix 6: Event History Creation and Documentation 319

NLSY97 Appendix 7: Continuous Month Scheme and Crosswalk 331

NLSY97 Appendix 8: Instrument Rosters 345

NLSY97 Appendix 9: Family Process and Adolescent Outcome Measures 371

NLSY97 Appendix 10: CAT-ASVAB Scores 377

NLSY97 Appendix 11: Collection of the Transcript Data, Wave 1 383

Although Appendix 9, "Family Process and Adolescent Outcome Measures," is considered part of the *Codebook Supplement*, it is provided in a separate printed document. The variables discussed in the appendix are listed in this document for user reference.

For more information about any aspect of the NLS program, contact:
NLS User Services
Center for Human Resource Research
921 Chatham Lane, Suite 100
Columbus, OH 43221-2418
(614) 442-7366
usersvc@postoffice.chrr.ohio-state.edu

Introduction to the CAPI Questionnaire and Codebook

The newest survey in the NLS program, the National Longitudinal Survey of Youth 1997 (NLSY97) is designed to be representative of the U.S. population born during the years 1980 through 1984. Through the NLSY97, the Bureau of Labor Statistics (BLS) can identify characteristics that define the transition today's youths make from school to the labor market and into adulthood. The NLSY97 cohort includes 8,984 respondents ages 12–16 as of December 31, 1996; the sample contains a cross-sectional subsample and an oversample of black and Hispanic respondents. More information on the selection of respondents for the cohort is available in the *NLSY97 User's Guide*.

This survey is conducted as a computer-assisted personal interview (CAPI). The Round 1 survey used three different questionnaires: the *Screener, Household Roster, and Non-Resident Roster Questionnaire;* the *Youth Questionnaire;* and the *Parent Questionnaire*. The *Screener, Household Roster, and Non-Resident Roster Questionnaire* was administered to a household resident over the age of 18. The *Youth Questionnaire* was administered to the youth respondent living in the household. The *Parent Questionnaire* was administered to a parent or parent-like figure of the youth residing in the household. For information about how the responding parent was selected, researchers should refer to the *NLSY97 User's Guide*.

The Rounds 2–4 surveys used a *Youth Questionnaire* similar to Round 1. These interviews also included a second instrument, a brief *Household Income Update* administered to one of the respondent's parents (if the respondent lived at home). No screener or parent interviews were conducted. The content of these survey instruments is presented in separate questionnaire documents, available from NLS User Services. More information about the administration of the various instruments is provided in the *NLSY97 User's Guide*.

Frequencies from the data collected during the NLSY97 interviews are provided for researchers in the form of a codebook contained on the NLSY97 CD-ROM. For each variable taken directly from the interview, the codebook provides information about the question or check item, the variable reference number, the universe of respondents to whom the question applied, and the distribution of responses to the question. The codebook also contains a number of variables created by survey personnel after the interview, providing information about the content of the variable and the distribution of responses. The figure below shows a sample codebook page for one of the NLSY97 variables.

Reference number

Question text

Response distribution

Universe information

```
R05151.00    [YEXP-2000]  PERCENT CHANCE R HAS COLLEGE DEGREE BY 30 YEARS OLD
                                                              Survey Year: 1997

Now think ahead to when you turn 30 years old.

What is the percent chance that you will have a four-year college degree by the
time you turn 30?

     185           0: 0%
      98           1 to 10: 1%-10%
      84          11 to 20: 11%-20%
     146          21 to 30: 21%-30%
      51          31 to 40: 31%-40%
     538          41 to 50: 41%-50%
      69          51 to 60: 51%-60%
      79          61 to 70: 61%-70%
     457          71 to 80: 71%-80%
     324          81 to 90: 81%-90%
    1480          91 to 100: 91%-100%
  -------
    3511

Refusal(-1)            2
Don't Know(-2)        23
Invalid Skip(-3)      29
TOTAL =========>    3565   Valid Skip(-4)    5419     Non-Interview(-5)       0

Min:       0        Mean:     72.78
Max:     100

Lead In: R05150.00[Default]
Default Next Question: R05152.00
```

Some information relevant to the data contained in the codebook cannot be included in the actual codebook documentation due to practical constraints. This document, the *NLSY97 Codebook Supplement,* presents additional information that will help researchers to use the data more effectively. The first attachment lists industry and occupation codes and their associated descriptors; this list is too lengthy to be incorporated into the codebook. A number of appendices then provide programs for the created variables; researchers may want to use this information to identify the raw survey data used in a created variable and to understand the rules applied during the creation process. This document also describes the creation of the event history arrays and helps researchers to understand their structure and use.

This introduction to the *Codebook Supplement* is intended to assist researchers in understanding some of the terms and survey methods associated with a CAPI interview. The way in which a CAPI instrument is constructed has important implications for the presentation of the data, and so this discussion will aid in interpretation of the codebook and accompanying documentation.

I. TERMS

Discussions of CAPI surveys include a number of terms that may be unfamiliar to many researchers. The following terms are used throughout the survey documentation; we provide definitions here for easy reference.

Instrument Rosters
A "roster" is a list of one or more items of information pertaining to a specific set of subjects, such as the biological children of the respondent or members of the respondent's household. For instance, the BIOCHILD roster contains a variety of items (e.g., name, gender, birthdate, etc.) pertaining to each biological child of the respondent. By using the roster format, the CAPI program can gather an inventory of information, tag the data to a specific subject, carry the data along through the interview, and access them when necessary. This format also allows for these items of information to be presented to the interviewer at any time. A listing of rosters, such as the Household Roster, is included in this *Codebook Supplement* as Appendix 8. This listing includes the contents of the rosters, the applicable names attached to the rosters that might be encountered in the codebook and/or the questionnaire, and the variable reference number assigned to each piece of data in the codebook.

Interviewer's Reference Manual
The *Interviewer's Reference Manual* reproduces the electronic "help screens" that were available to the interviewers for specific questions. These help screens contain definitions, instructions, and other information which interviewers used during the interview to obtain consistent information from respondents, as well as listings of the questions for which they were used during the interview. The *Interviewer's Reference Manual* should be used in conjunction with the codebook or the questionnaire so that researchers can fully understand the intent of each question in the survey.

Symbols and "Text Fills"
Symbols are reserved fields into which data can be placed, stored, and accessed throughout the questionnaire. Each symbol field is assigned a distinct name, often with an index number appended when the same piece of information is stored for a set of subjects. For instance, the symbol "lintdate" contains the date of last interview for the respondent. The symbols "emp.name(1)" through "emp.name(7)" contain the names of the first through the seventh employers of the respondent. Throughout the survey, the last interview date and/or the name of the respondent's first employer can be accessed by invoking the symbol names "lintdate" and "emp.name(1)" respectively.

Users will encounter these symbol names both in the codebook and in the questionnaire. Sometimes data in a symbol is accessed and used to govern a skip or perform a calculation. At other times, a symbol name may be part of a question text. In this case, the content of the specific symbol content becomes part of the text of the

question, and is read as such. This is referred to as a "text substitution" or "text fill." For instance, a question text reading "When you started working for [emp.name(1)], what kind of work did you do? That is, what was your occupation?" would appear to the interviewer with the appropriate name replacing the symbol "emp.name(1)."

Other types of text fills are also present in the questionnaire. For example, the responding parent may be asked the following question: "What was the reason [he/she] did not live with [his/her] [mother/father/ parents]?" In this question, "he/she" and "his/her" refer to the parent's spouse or partner. The computer automatically fills in the correct gender, and the interviewer reads the question appropriately. "Mother/ father/parents" in the question text indicates a set of possible responses to the previous question. The computer automatically fills in the appropriate choice from the three words for the given interview. These automated text fills reduce error and remove the burden of asking the question using the appropriate phrasing from the interviewer.

Loops

Certain sequences of questions in the CAPI instrument are repeated a number of times. For instance, in round 1 some sets of questions were repeated up to 7 times in the Employment Section of the youth questionnaire, for each of up to 7 jobs. Question names that include a ".01", ".02", ".03", etc. at the end, belong to these repeating sequences of questions. Each repetition of the sequence of questions is referred to as a "loop." Taking the Employment Section as an example, the sequence of questions asking about the first job is referred to as the first "loop." The sequence asking about the second employer is referred to as the second "loop," and so on. While the codebook generally includes more than one loop in a series (because more than one may contain valid data), the questionnaire includes only the **first loop** in each set of loops.

Hard and Soft Range Restrictions

The "Hard Minimum," "Hard Maximum," "Soft Minimum," and "Soft Maximum" specifications control the allowable range of values that can be entered for a given question. These fields are only active when a question calls for the entry of a time date or amount. Questions that require the interviewer to select one response or to select all that apply do not contain this field, as the range limits are implicit in the distribution code block and are thereby enforced.

Hard minima and maxima are absolute limits that an interviewer- or respondent-generated answer must obey. Entry of values outside the hard range is not allowed. In such cases the interviewer is instructed to enter the maximum or minimum allowable value, as appropriate, enter the actual response in the comment field, and "flag" the case for central office checking. Soft minima and maxima are nested within the hard range. When a response falls outside the soft range but inside the hard range, the computer beeps and asks the interviewer to either confirm or change the response.

In some cases, the hard ranges are themselves determined by a variable. For example, a question may use the respondent's birth date as a minimum and the current interview date as a maximum, preventing the interviewer from entering a date earlier than the hard minimum birth date or later than the hard maximum interview date. This is a powerful tool for the collection of event histories (among other types of data sequences) and is used extensively in the instrument.

II. Appearance and Presentation of Data

Some CAPI questions generate more than one variable. For example, some questions collect information about the date an event happened and generate three variables: month, day, and year. Similarly, questions that ask the respondent to indicate which of several responses are appropriate, and to pick all responses that apply, can generate multiple variables. When a question record generates multiple variables, those variables have decimals in the reference numbers. These two types of questions (date-entry and "code-all-that-apply") and the examples of the resulting variables, are discussed further below.

Date-Entry Questions

All date questions, whether full dates (month, day and year) or just month and year, are represented in the CAPI codebook with, respectively, three or two variables. Each variable contains the same codeblock displaying the ranges and missing values for all elements of the date. A base reference number ending in ".00", is assigned to the first variable in the set. The same base reference number, with endings of ".01" and ".02" if necessary, is assigned to the other elements of the date. Thus, if the day is assigned a reference number of R10851.00 for the variable "Date of Birth of HH Member 01 (Scr Ros Item)," the month and year would be assigned reference numbers of R10851.01 and R10851.02 respectively.

Code-All-That-Apply Questions

The NLSY97 includes questions that allow respondents to give multiple responses or code all responses that apply. In the CAPI codebook, each possible response constitutes its own variable. In each codeblock for a given code-all-that-apply question, frequencies for **all** possible responses are represented. However, each specific variable contains **only** the valid data for one specific possible response, as researchers will note when extracting data. Reference numbers are assigned in the same manner as described for date-entry questions. A base reference number is assigned to the first possible response, with a decimal value being appended to that base number for each following possible response. For example, variables R02461.00–R02461.11 (Activities to Find a Job Emp 01) provide responses (coded yes/no) for 12 different methods of finding a job. Although the codeblock for R02461.01 represents the frequencies for all possible responses to the question, accessing the data for that specific variable will produce only the data for the response "Contacted employment agency."

Machine-Generated Check Items

Many questions in the NLSY97 data are "machine checks." In these questions, previously reported information is checked by the computer and computations are made automatically, causing the appropriate skip pattern to be executed without intervention by the interviewer. For example, the survey program may check the respondent's gender before asking women whether they have ever been pregnant and men whether they have ever fathered a child. In an effort to clarify the skip patterns present in the instrument, a large number of these machine checks appear in the codebook. The text of machine checks is generally in machine language or equation form; the codeblock also includes a note clarifying the purpose of the check. For example, R02678., "Chk R Female (Pregnancy Leave) Emp 01," includes the following code:

```
([male/female] = 2);
```

The codeblock also contains a translation of the code that is more accessible to users:

```
/* Is respondent a female? This determines whether the paid pregnancy leave questions which follow
are asked. */
```

NLSY97 Attachment 1:

1990 Census Industrial & Occupational Classification Codes

U.S. DEPARTMENT OF COMMERCE
Bureau of the Census
Washington, DC 20233

1990 CENSUS OF POPULATION INDUSTRIAL CLASSIFICATION SYSTEM

"N.e.c." means not elsewhere classified.

1990 Census Code	Industry
	AGRICULTURE, FORESTRY, AND FISHERIES
010	Agricultural production, crops
011	Agricultural production livestock
012	Veterinary services
020	Landscape and horticultural services
030	Agricultural services, n.e.c.
031	Forestry
032	Fishing, hunting, and trapping
	MINING
040	Metal mining
041	Coal mining
042	Oil and gas extraction
050	Nonmetallic mining and quarrying, except fuel
060	**CONSTRUCTION**
	MANUFACTURING
	Nondurable Goods
	Food and kindred products
100	Meat products
101	Dairy products
102	Canned, frozen, and preserved fruits and vegetables
110	Grain mill products
111	Bakery products
112	Sugar and confectionery products
120	Beverage industries
121	Miscellaneous food preparations and kindred products
122	Not specified food industries
130	Tobacco manufactures
	Textile mill products
132	Knitting mills
140	Dyeing and finishing textiles, except wool and knit goods
141	Carpets and rugs
142	Yarn, thread, and fabric mills
150	Miscellaneous textile mill products
	Apparel and other finished textile products
151	Apparel and accessories, except knit
152	Miscellaneous fabricated textile products
	Paper and allied products
160	Pulp, paper, and paperboard mills
161	Miscellaneous paper and pulp products
162	Paperboard containers and boxes

1990 Census Code	Industry
	Printing, publishing, and allied industries
171	Newspaper publishing and printing
172	Printing, publishing and allied industries, except newspapers
	Chemicals and allied products
180	Plastics, synthetics, and resins
181	Drugs
182	Soaps and cosmetics
190	Paints, varnishes, and related products
191	Agricultural chemicals
192	Industrial and miscellaneous chemicals
	Petroleum and coal products
200	Petroleum refining
201	Miscellaneous petroleum and coal products
	Rubber and miscellaneous plastics products
210	Tires and inner tubes
211	Other rubber products, and plastics footwear and belting
212	Miscellaneous plastics products
	Leather and leather products
220	Leather tanning and finishing
221	Footwear, except rubber and plastic
222	Leather products, except footwear
	Durable Goods
	Lumber and wood products, except furniture
230	Logging
231	Sawmills, planing mills, and millwork
232	Wood buildings and mobile homes
241	Miscellaneous wood products
242	Furniture and fixtures
	Stone, clay, glass, and concrete products
250	Glass and glass products
251	Cement, concrete, gypsum, and planter products
252	Structural clay products
261	Pottery and related products
262	Miscellaneous nonmetallic mineral and stone products
	Metal industries
270	Blast furnaces, steelworks, rolling and finishing mills
271	Iron and steel foundries
272	Primary aluminum industries
280	Other primary metal industries
281	Cutlery, handtools, and general hardware
282	Fabricated structural metal products
291	Metal forgings and stampings
292	Ordnance
300	Miscellaneous fabricated metal products
301	Not specified metal industries

1990 Census Code	Industry
	Machinery and computing equipment
310	Engines and turbines
311	Farm machinery and equipment
312	Construction and material handling machines
320	Metalworking machinery
321	Office and accounting machines
322	Computers and related equipment
331	Machine, except electrical, n.e.c.
332	Not specified machinery
	Electrical machinery, equipment, and supplies
340	Household appliances
341	Radio, TV, and communication equipment
342	Electrical machinery, equipment, and supplies, n.e.c.
350	Not specified electrical machinery, equipment, and supplies
	Transportation equipment
351	Motor vehicles and motor vehicle equipment
352	Aircraft and parts
360	Ship and boat building and repairing
361	Railroad locomotives and equipment
362	Guided missiles, space vehicles, and parts
370	Cycles and miscellaneous transportation equipment
	Professional and photographic equipment, and watches
371	Scientific and controlling instruments
372	Medical, dental, and optical instruments and supplies
380	Photographic equipment and supplies
381	Watches, clocks, and clockwork operated devices
390	Toys, amusement, and sporting goods
391	Miscellaneous manufacturing industries
392	Not specified manufacturing industries
	TRANSPORTATION, COMMUNICATIONS AND OTHER PUBLIC UTILITIES
	Transportation
400	Railroads
401	Bus service and urban transit
402	Taxicab service
410	Trucking service
411	Warehousing and storage
412	U.S. Postal Service
420	Water transportation
421	Air transportation
422	Pipelines, except natural gas
432	Services incidental to transportation
	Communications
440	Radio and television broadcasting and cable
441	Telephone communications
442	Telegraph and miscellaneous communications services

1990 Census Code	Industry
	Utilities and sanitary services
450	Electric light and power
451	Gas and steam supply systems
452	Electric and gas, and other combinations
470	Water supply and irrigation
471	Sanitary services
472	Not specified utilities
	WHOLESALE TRADE
	Durable Goods
500	Motor vehicles and equipment
501	Furniture and home furnishings
502	Lumber and construction materials
510	Professional and commercial equipment and supplies
511	Metals and minerals, except petroleum
512	Electrical goods
521	Hardware, plumbing and heating supplies
530	Machinery, equipment, and supplies
531	Scrap and waste materials
532	Miscellaneous wholesale, durable goods
	Nondurable Goods
540	Paper and paper products
541	Drugs, chemicals and allied products
542	Apparel, fabrics, and notions
550	Groceries and related products
551	Farm-product raw materials
552	Petroleum products
560	Alcoholic beverages
561	Farm supplies
562	Miscellaneous wholesale, nondurable goods
571	Not specified wholesale trade
	RETAIL TRADE
580	Lumber and building material retailing
581	Hardware stores
582	Retail nurseries and garden stores
590	Mobile home dealers
591	Department stores
592	Variety stores
600	Miscellaneous general merchandise stores
601	Grocery stores
602	Dairy products stores
610	Retail bakeries
611	Food stores, n.e.c.
612	Motor vehicle dealers
620	Auto and home supply stores
621	Gasoline service stations
622	Miscellaneous vehicle dealers
623	Apparel and accessory stores, except shoe
630	Shoe stores
631	Furniture and home furnishings stores

1990 Census Code	Industry
632	Household appliance stores
633	Radio, TV, and computer stores
640	Music stores
641	Eating and drinking places
642	Drug stores
650	Liquor stores
651	Sporting goods, bicycles, and hobby stores
652	Book and stationary stores
660	Jewelry stores
661	Gift, novelty, and souvenir shops
662	Sewing, needlework and piece goods stores
663	Catalog and mail order houses
670	Vending machine operators
671	Direct selling establishments
672	Fuel dealers
681	Retail florists
682	Miscellaneous retail stores
691	Not specified retail trade
	FINANCE, INSURANCE, AND REAL ESTATE
700	Banking
701	Savings institutions, including credit unions
702	Credit agencies, n.e.c.
710	Security, commodity brokerage, and investment companies
711	Insurance
712	Real estate, including real estate-insurance offices
	BUSINESS AND REPAIR SERVICES
721	Advertising
722	Services to dwellings and other buildings
731	Personnel supply services
732	Computer and data processing services
740	Detective and protective services
741	Business services, n.e.c.
742	Automotive rental and leasing, without drivers
750	Automobile parking and carwashes
751	Automotive repair and related services
752	Electrical repair shops
760	Miscellaneous repair services
	PERSONAL SERVICES
761	Private households
762	Hotels and motels
770	Lodging places, except hotels and motels
771	Laundry, cleaning, and garment services
772	Beauty shops
780	Barber shops
781	Funeral service and crematories
782	Shoe repair shops
790	Dressmaking shops
791	Miscellaneous personal services

1990 Census Code	Industry
	ENTERTAINMENT AND RECREATION SERVICES
800	Theaters and motion pictures
801	Video tape rental
802	Bowling centers
810	Miscellaneous entertainment and recreation services
	PROFESSIONAL AND RELATED SERVICES
812	Offices and clinics of physicians
820	Offices and clinics of dentists
821	Offices and clinics of chiropractors
822	Offices and clinics of optometrists
830	Offices and clinics of health practitioners, n.e.c.
831	Hospitals
832	Nursing and personal care facilities
840	Health services, n.e.c.
841	Legal services
842	Elementary and secondary schools
850	Colleges and universities
851	Vocational schools
852	Libraries
860	Educational services, n.e.c.
861	Job training and vocational rehabilitation services
862	Child day care services
863	Family child care homes
870	Residential care facilities, without nursing
871	Social services, n.e.c.
872	Museums, art galleries and zoos
873	Labor unions
880	Religious organizations
881	Membership organizations, n.e.c.
882	Engineering architectural and surveying services
890	Accounting, auditing, and bookkeeping services
891	Research, development, and testing services
892	Management and public relations services
893	Miscellaneous professional and related services
	PUBLIC ADMINISTRATION
900	Executive and legislative offices
901	General government, n.e.c.
910	Justice, public order, and safety
921	Public finance, taxation, and monetary policy
922	Administration of human resources programs
930	Administration of environmental quality and housing programs
931	Administration of economic programs
932	National security and international affairs
940	Army
941	Air Force
942	Navy
950	Marines
991	Military

U.S. DEPARTMENT OF COMMERCE
Bureau of the Census
Washington, DC 20233

1990 CENSUS OF POPULATION OCCUPATIONAL CLASSIFICATION SYSTEM

"N.e.c." means not elsewhere classified.

1990 Census Code	Occupation
	MANAGERIAL AND PROFESSIONAL SPECIALTY OCCUPATIONS
	Executive, Administrative, and Managerial Occupations
003	Legislators
004	Chief executives and general administrators, public administration
005	Administrators and officials, public administration
006	Administrators, protective services
007	Financial managers
008	Personnel and labor relations managers
009	Purchasing managers
013	Managers, marketing, advertising, and public relations
014	Administrators, education and related fields
015	Managers, medicine and health
016	Postmasters and mail superintendents
017	Managers, food serving and lodging establishments
018	Managers, properties and real estate
019	Funeral directors
021	Managers, service organizations, n.e.c.
022	Managers and administrators, n.e.c.
	Management Related Occupations
023	Accountants and auditors
024	Underwriters
025	Other financial officers
026	Management analysts
027	Personnel, training, and labor relations specialists
028	Purchasing agents and buyers, farm products
029	Buyers, wholesale and retail trade except farm products
033	Purchasing agents and buyers, n.e.c.
034	Business and promotion agents
035	Construction inspectors
036	Inspectors and compliance officers, except construction
037	Management related occupations, n.e.c.
	PROFESSIONAL SPECIALTY OCCUPATIONS
	Engineers, Architects, and Surveyors
043	Architects
	Engineers
044	Aerospace
045	Metallurgical and materials
046	Mining
047	Petroleum
048	Chemical
049	Nuclear
053	Civil

1990 Census Code	Occupation
054	Agricultural
055	Electrical and electronic
056	Industrial
057	Mechanical
058	Marine and naval architects
059	Engineers, n.e.c.
063	Surveyors and mapping scientists
	Mathematical and Computer Scientists
064	Computer systems analysts and scientists
065	Operations and systems researchers and analysts
066	Actuaries
067	Statisticians
068	Mathematical scientists, n.e.c.
	Natural Scientists
069	Physicists and astronomers
073	Chemists, except biochemists
074	Atmospheric and space scientists
075	Geologists and geodesists
076	Physical scientists, n.e.c.
077	Agricultural and food scientists
078	Biological and life scientists
079	Forestry and conservation scientists
083	Medical scientists
	Health Diagnosing Occupations
084	Physicians
085	Dentists
086	Veterinarians
087	Optometrists
088	Podiatrists
089	Health diagnosing practitioners, n.e.c.
	Health Assessment and Treating Occupations
095	Registered nurses
096	Pharmacists
097	Dietitians
	Therapists
098	Respiratory therapists
099	Occupational therapists
103	Physical therapists
104	Speech therapists
105	Therapists, n.e.c.
106	Physicians' assistants
	Teachers, Postsecondary
113	Earth, environmental, and marine science teachers
114	Biological science teachers
115	Chemistry teachers

1990 Census Code	Occupation
116	Physics teachers
117	Natural science teachers, n.e.c.
118	Psychology teachers
119	Economics teachers
123	History teachers
124	Political science teachers
125	Sociology teachers
126	Social science teachers, n.e.c.
127	Engineering teachers
128	Mathematical science teachers
129	Computer science teachers
133	Medical science teachers
134	Health specialties teachers
135	Business, commerce, and marketing teachers
136	Agriculture and forestry teachers
137	Art, drama, and music teachers
138	Physical education teachers
139	Education teachers
143	English teachers
144	Foreign language teachers
145	Law teachers
146	Social work teachers
147	Theology teachers
148	Trade and industrial teachers
149	Home economics teachers
153	Teachers, postsecondary, n.e.c.
154	Postsecondary teachers, subject not specified
	Teachers, Except Postsecondary
155	Teachers, prekindergarten and kindergarten
156	Teachers, elementary school
157	Teachers, secondary school
158	Teachers, special education
159	Teachers, n.e.c.
163	Counselors, educational and vocational
	Librarians, Archivists, and Curators
164	Librarians
165	Archivists and curators
	Social Scientists and Urban Planners
166	Economists
167	Psychologists
168	Sociologists
169	Social scientists, n.e.c.
173	Urban planners
	Social, Recreation, and Religious Workers
174	Social workers
175	Recreation workers
176	Clergy

1990 Census Code	Occupation
177	Religious workers, n.e.c.
	Lawyers and Judges
178	Lawyers
179	Judges
	Writers, Artists, Entertainers, and Athletes
183	Authors
184	Technical writers
185	Designers
186	Musicians and composers
187	Actors and directors
188	Painters, sculptors, craft-artists, and artist printmakers
189	Photographers
193	Dancers
194	Artists, performers, and related workers, n.e.c.
195	Editors and reporters
197	Public relations specialists
198	Announcers
199	Athletes
	TECHNICAL, SALES AND ADMINISTRATIVE SUPPORT OCCUPATIONS
	Technicians and Related Support occupations
	Health Technologists and Technicians
203	Clinical laboratory technologists and technicians
204	Dental hygienists
205	Health record technologists and technicians
206	Radiologic technicians
207	Licensed practical nurses
208	Health technologists and technicians, n.e.c.
	Technologists and Technicians, Except Health
	Engineering and Related Technologists and Technicians
213	Electrical and electronic technicians
214	Industrial engineering technicians
215	Mechanical engineering technicians
216	Engineering technicians, n.e.c.
217	Drafting occupations
218	Surveying and mapping technicians
	Science Technicians
223	Biological technicians
224	Chemical technicians
225	Science technicians
	Technicians: Except Health, Engineering and Science
226	Airplane pilots and navigators
227	Air traffic controllers
228	Broadcast equipment operators
229	Computer programmers
233	Tool programmers, numerical control
234	Legal assistants

1990 Census Code	Occupation
235	Technicians, n.e.c.
	Sales Occupations
243	Supervisors and proprietors, sales occupations
	Sales Representatives, Finance and Business Services
253	Insurance sales occupations
254	Real estate sales occupations
255	Securities and financial services sales occupations
256	Advertising and related sales occupations
257	Sales occupations, other business services
	Sales Representatives, Commodities Except Retail
258	Sales engineers
259	Sales representatives, mining, manufacturing, and wholesale
	Sales Workers, Retail and Personal Services
263	Sales workers, motor vehicles and boats
264	Sales workers, apparel
265	Sales workers, shoes
266	Sales workers, furniture and home furnishings
267	Sales workers; radio, TV, hi-fi, and appliances
268	Sales workers, hardware and building supplies
269	Sales workers, parts
274	Sales workers, other commodities
275	Sales counter clerks
276	Cashiers
277	Street and door-to-door sales workers
278	News vendors
	Sales Related Occupations
283	Demonstrators, promoters and models, sales
284	Auctioneers
285	Sales support occupations, n.e.c.
	Administrative Support Occupations, Including Clerical
	Supervisors, Administrative Support Occupations
303	Supervisors, general office
304	Supervisors, computer equipment operators
305	Supervisors, financial records processing
306	Chief communications operators
307	Supervisors; distribution, scheduling, and adjusting clerks
	Computer Equipment Operators
308	Computer operators
309	Peripheral equipment operators
	Secretaries, Stenographers, and Typists
313	Secretaries
314	Stenographers
315	Typists
	Information Clerks

1990 Census Code	Occupation
316	Interviewers
317	Hotel clerks
318	Transportation ticket and reservation agents
319	Receptionists
323	Information clerks, n.e.c.
	Records Processing Occupations, Except Financial
325	Classified-ad clerks
326	Correspondence clerks
327	Order clerks
328	Personnel clerks, except payroll and timekeeping
329	Library clerks
335	File clerks
336	Records clerks
	Financial Records Processing Occupations
337	Bookkeepers, accounting, and auditing clerks
338	Payroll and timekeeping clerks
339	Billing clerks
343	Cost and rate clerks
344	Billing, posting, and calculating machine operators
	Duplicating, Mail and Other Office Machine Operators
345	Duplicating machine operators
346	Mail preparing and paper handling machine operators
347	Office machine operators, n.e.c.
	Communications Equipment Operators
348	Telephone operators
353	Communications equipment operators, n.e.c.
	Mail and Message Distributing Occupations
354	Postal clerks, exc. mail carriers
355	Mail carriers, postal service
356	Mail clerks, exc. postal service
357	Messengers
	Material Recording, Scheduling, and Distributing Clerks
359	Dispatchers
363	Production coordinators
364	Traffic, shipping, and receiving clerks
365	Stock and inventory clerks
366	Motor readers
368	Weighers, measurers, checkers and samplers
373	Expediters
374	Material recording, scheduling, and distributing clerks, n.e.c.
	Adjusters and Investigators
375	Insurance adjusters, examiners, and investigators
376	Investigators and adjusters except insurance
377	Eligibility clerks, social welfare
378	Bill and account collectors

1990 Census Code	Occupation
	Miscellaneous Administrative Support Occupations
379	General office clerks
383	Bank tellers
384	Proofreaders
385	Data-entry keyers
386	Statistical clerks
387	Teachers' aides
389	Administrative support occupations, n.e.c.
	SERVICE OCCUPATIONS
	Private Household Occupations
403	Launderers and ironers
404	Cooks, private household
405	Housekeepers and butlers
406	Child care workers, private household
407	Private household cleaners and servants
	Protective Service Occupations
	Supervisors Protective Service Occupations
413	Supervisors, firefighting and fire prevention occupations
414	Supervisors, police and detectives
415	Supervisors, guards
	Firefighting and Fire Prevention Occupations
416	Fire inspection and fire prevention occupations
417	Firefighting occupations
	Police and Detectives
418	Police and detectives, public service
423	Sheriffs, bailiffs, and other law enforcement officers
424	Correctional institution officers
	Guards
425	Crossing guards
426	Guards and police, except public service
427	Protective service occupations, n.e.c.
	Service Occupations, Except Protective and Household
	Food Preparation and Service Occupations
433	Supervisors, food preparation and service occupations
434	Bartenders
435	Waiters and waitresses
436	Cooks
438	Food counter, fountain and related occupations
439	Kitchen workers, food preparation
443	Waiters'/waitresses' assistants
444	Miscellaneous food preparation occupations
	Health Service Occupations
445	Dental assistants
446	Health aides, except nursing

1990 Census Code	Occupation
447	Nursing aides, orderlies, and attendants
	Cleaning and Building Service Occupations, except Household
448	Supervisors cleaning and building service workers
449	Maids and houseman
453	Janitors and cleaners
454	Elevator operators
455	Post control occupations
	Personal Service Occupations
456	Supervisors, personal service occupations
457	Barbers
458	Hairdressers and cosmetologists
459	Attendants, amusement and recreation facilities
461	Guides
462	Ushers
463	Public transportation attendants
464	Baggage porters and bellhops
465	Welfare service aides
466	Family child care providers
467	Early childhood teacher's assistants
468	Child care workers, n.e.c.
469	Personal service occupations, n.e.c.
	FARMING, FORESTRY AND FISHING OCCUPATIONS
	Farm Operators and Managers
473	Farmers, except horticultural
474	Horticultural specialty farmers
475	Managers, farms, except horticultural
476	Managers, horticultural specialty farms
	Other Agricultural and Related Occupations
	Farm Occupations Except Managerial
477	Supervisors, farm workers
479	Farm workers
483	Marine life cultivation workers
484	Nursery workers
	Related Agricultural Occupations
485	Supervisors related agricultural occupations
486	Groundskeepers and gardeners, except farm
487	Animal caretakers, except farm
488	Graders and sorters, agricultural products
489	Inspectors, agricultural products
	Forestry and Logging Occupations
494	Supervisors, forestry, and logging workers
495	Forestry workers, except logging
496	Timber cutting and logging occupations
	Fishers, Hunters, and Trappers
497	Captains and other officers, fishing vessels

1990 Census Code	Occupation
498	Fishers
499	Hunters and trappers
	PRECISION PRODUCTION, CRAFT, AND REPAIR OCCUPATIONS
	Mechanics and Repairers
503	Supervisors, mechanics and repairers
	Mechanics and Repairers, Except Supervisors
	Vehicle and Mobile Equipment Mechanics and Repairers
505	Automobile mechanics
506	Automobile mechanic apprentices
507	Bus, truck, and stationary engine mechanics
508	Aircraft engine mechanics
509	Small engine repairers
514	Automobile body and related repairers
515	Aircraft mechanics, exc. engine
516	Heavy equipment mechanics
517	Farm equipment mechanics
518	Industrial machinery repairers
519	Machinery maintenance occupations
	Electrical and Electronic Equipment Repairers
523	Electronic repairers, communications and industrial equipment
525	Data processing equipment repairers
526	Household appliance and power tool repairers
527	Telephone line installers and repairers
529	Telephone installers and repairers
533	Miscellaneous electrical and electronic equipment repairers
534	Heating, air conditioning, and refrigeration mechanics
	Miscellaneous Mechanics and Repairers
535	Camera, watch, and musical instrument repairers
536	Locksmiths and safe repairers
538	Office machine repairers
539	Mechanical controls and valve repairers
543	Elevator installers and repairers
544	Millwrights
547	Specified mechanics and repairers, n.e.c.
549	Not specified mechanics and repairers
	Construction Trades
	Supervisors, Construction Occupations
553	Supervisors; brickmasons, stonemasons, and tile setters
554	Supervisors; carpenters and related workers
555	Supervisors; electricians and power transmission installers
556	Supervisors; painters, paperhangers, and plasterers
557	Supervisors; plumbers, pipefitters, and steamfitters
558	Supervisors; n.e.c.
	Construction Trades Except Supervisors
563	Brickmasons and stonemasons
564	Brickmason and stonemason apprentices

1990 Census Code	Occupation
565	Tile setters, hard and soft
566	Carpet installers
567	Carpenters
569	Carpenter apprentices
573	Drywall installers
575	Electricians
576	Electrician apprentices
577	Electrical power installers and repairers
579	Painters, construction and maintenance
583	Paperhangers
584	Plasterers
585	Plumbers, pipefitters, and steamfitters
587	Plumber, pipefitter, and steamfitter apprentices
588	Concrete and terrazzo finishers
589	Glaziers
593	Insulation workers
594	Paving, surfacing, and tamping equipment operators
595	Roofers
596	Sheetmetal duct installers
597	Structural metal workers
598	Drillers, earth
599	Construction trades, n.e.c.
	Extractive Occupations
613	Supervisors, extractive occupations
614	Drillers, oil well
615	Explosives workers
616	Mining machine operators
617	Mining occupations, n.e.c.
	Precision Production Occupations
628	Supervisors, production occupations
	Precision Metal Working Occupations
634	Tool and die makers
635	Tool and die maker apprentices
636	Precision assemblers, metal
637	Machinists
639	Machinist apprentices
643	Boilermakers
644	Precision grinders, filers, and tool sharpeners
645	Patternmakers and model makers, metal
646	Lay-out workers
647	Precious stones and metals workers (Jewelers)
649	Engravers, metal
653	Sheet metal workers
654	Sheet metal worker apprentices
655	Miscellaneous precision metal workers
	Precision Woodworking Occupations
656	Patternmakers and model makers, wood
657	Cabinet makers and bench carpenters

1990 Census Code	Occupation
658	Furniture and wood finishers
659	Miscellaneous precision woodworkers
	Precision Textile, Apparel, and Furnishings Machine Workers
666	Dressmakers
667	Tailors
668	Upholsterers
669	Shoe repairers
674	Miscellaneous precision apparel and fabric workers
	Precision Workers, Assorted Materials
675	Hand molders and shapers, except jewelers
676	Patternmakers, lay-out workers, and cutters
677	Optical goods workers
678	Dental laboratory and medical appliance technicians
679	Bookbinders
683	Electrical and electronic equipment assemblers
684	Miscellaneous precision workers, n.e.c.
	Precision Food Production Occupations
686	Butchers and meat cutters
687	Bakers
688	Food batchmakers
	Precision Inspectors, Testers, and Related Workers
689	Inspectors, testers, and graders
693	Adjusters and calibrators
	Plant and System Operators
694	Water and sewage treatment plant operators
695	Power plant operators
696	Stationary engineers
699	Miscellaneous plant and system operators
	OPERATORS, FABRICATORS, AND LABORERS
	Machine Operators, Assemblers, and Inspectors
	Machine Operators and Tenders, except Precision Metal Working and Plastic Working Machine Operators
703	Lathe and turning machine set-up operators
704	Lathe and turning machine operators
705	Milling and planing machine operators
706	Punching and stamping press machine operators
707	Rolling machine operators
708	Drilling and boring machine operators
709	Grinding, abrading, buffing, and polishing machine operators
713	Forging machine operators
714	Numerical control machine operators
715	Miscellaneous metal, plastic, stone, and glass working machine operators
717	Fabricating machine operators, n.e.c.
	Metal and Plastic Processing Machine Operators
719	Molding and canting machine operators

1990 Census Code	Occupation
723	Metal plating machine operators
724	Heat treating equipment operators
725	Miscellaneous metal and plastic processing machine operators
	Woodworking Machine Operators
726	Wood lathe, routing, and planing machine operators
727	Sawing machine operators
728	Shaping and joining machine operators
729	Nailing and tacking machine operators
733	Miscellaneous woodworking machine operators
	Printing Machine Operators
734	Printing press operators
735	Photoengravers and lithographers
736	Typesetters and compositors
737	Miscellaneous printing machine operators
	Textile, Apparel, and Furnishings Machine Operators
738	Winding and twisting machine operators
739	Knitting, looping, taping, and weaving machine operators
743	Textile cutting machine operators
744	Textile serving machine operators
745	Shoe machine operators
747	Pressing machine operators
748	Laundering and dry cleaning machine operators
749	Miscellaneous textile machine operators
	Machine Operators, Assorted Materials
753	Cementing and gluing machine operators
754	Packaging and filling machine operators
755	Extruding and forming machine operators
756	Mixing and blending machine operators
757	Separating, filtering, and clarifying machine operators
758	Compressing and compacting machine operators
759	Painting and paint spraying machine operators
763	Roasting and baking machine operators, food
764	Washing, cleaning, and pickling machine operators
765	Folding machine operators
766	Furnace, kiln, and oven operators, exc. food
768	Crushing and grinding machine operators
769	Slicing and cutting machine operators
773	Motion picture projectionists
774	Photographic process machine operators
777	Miscellaneous machine operators, n.e.c.
779	Machine operators, not specified
	Fabricators, Assemblers, and Hand Working Occupations
783	Welders and cutters
784	Solderers and brazers
785	Assemblers
786	Hand cutting and trimming occupations
787	Hand molding, casting, and forming occupations

1990 Census Code	Occupation
789	Hand painting, coating, and decorating occupations
793	Hand engraving and printing occupations
795	Miscellaneous hand working occupations
	Production Inspectors, Testers, Samplers, and Weighers
796	Production inspectors, checkers, and examiners
797	Production testers
798	Production samplers and weighers
799	Graders and sorters, exc. agricultural
	Transportation and Material Moving Occupations
	Motor Vehicle Operators
803	Supervisors, motor vehicle operators
804	Truck drivers
806	Driver-sales workers
808	Bus drivers
809	Taxicab drivers and chauffeurs
813	Parking lot attendants
814	Motor transportation occupations, n.e.c.
	Transportation Occupations, Except Motor Vehicles
	Rail Transportation Occupations
823	Railroad conductors and yardmasters
824	Locomotive operating occupations
825	Railroad brake, signal, and switch operators
826	Rail vehicle operators, n.e.c.
	Water Transportation Occupations
828	Ship captains and mates, except fishing boats
829	Sailors and deckhands
833	Marine engineers
834	Bridge, lock, and lighthouse tenders
	Material Moving Equipment Operators
843	Supervisors, material moving equipment operators
844	Operating engineers
845	Longshore equipment operators
848	Hoist and winch operators
849	Crane and tower operators
853	Excavating and loading machine operators
855	Grader, dozer, and scraper operators
856	Industrial truck and tractor equipment operators
859	Miscellaneous material moving equipment operators
	Handlers, Equipment Cleaners, Helpers, and Laborers
864	Supervisors, handlers, equipment cleaners, and laborers, n.e.c.
865	Helpers, mechanics and repairers
	Helpers, Construction and Extractive Occupations
866	Helpers, construction trades
867	Helpers, surveyor
868	Helpers, extractive occupations

1990 Census Code	Occupation
869	Construction laborers
874	Production helpers
	Freight, Stock, and Material Handlers
875	Garbage collectors
876	Stevedores
877	Stock handlers and baggers
878	Machine feeders and offbearers
883	Freight, stock, and material handlers, n.e.c.
885	Garage and service station related occupations
887	Vehicle washers and equipment cleaners
888	Hand packers and packagers
889	Laborers, except construction
	MILITARY OCCUPATIONS
905	Military occupation

Researchers should be aware of several issues regarding military industries and occupations. The questionnaire is designed so that respondents in the active Armed Forces should not be asked to report the industry and occupation of that job in the same questions that collect information about civilian jobs. Rather, respondents report their military occupational specialty as part of a series tailored to military jobs that also asks about the branch of the Armed Forces, the respondent's dates of enlistment, military pay grade, etc. This means that the military industry and occupation codes should appear very rarely in the NLSY97 data, occurring only if a respondent has a civilian job but is associated with the military in some way. However, some respondents in the active military answered the civilian jobs questions instead of the military questions in rounds 2 and 3, causing military respondents to be included in the industry and occupation variables as well as other regular employment questions like hourly rate of pay. This occurred for two reasons.

First, the question that identifies military jobs is YEMP-58500, the class of worker question, which asks whether the respondent's employer was the government, a private employer, a non-profit organization, a family business or farm, or the active Armed Forces. Respondents in the military should respond that they are in the active Armed Forces, and they are then asked the series of questions tailored to military jobs. However, some respondents with military jobs appear to have answered this question incorrectly; most of these respondents reported that they worked for the government. These respondents were then asked the set of questions about regular civilian employee jobs.

Second, some military respondents in round 3 were incorrectly routed through the civilian jobs section due to a programming error. If a respondent states that he or she is still employed with an employer reported in a previous round, the class of worker question is not asked. Instead, the respondent is routed to the correct series of questions based on a preset flag that identifies the job as military or civilian. However, this flag was not set correctly for respondents who reported military employment in round 2, and so these respondents were routed to the civilian series of questions in round 3.

These problems affected a very small number of respondents in rounds 2 and 3. However, researchers interested in civilian employment may want to remove these respondents from their analyses; conversely, users investigating military employment may want to include these respondents even though they did not answer the military series of questions. Researchers can identify respondents in the active Armed Forces by examining the variables MILFLAG and MILCODE on the employer (YEMP) roster. These roster items were edited to identify military jobs regardless of whether they were reported in the military or civilian section of the instrument. Virtually all military jobs appear as job #01 on the roster. Information about using roster variables in analyses is provided in Appendix 8 in this *Codebook Supplement*.

Introduction to the Created Variable Appendices

The main file NLSY97 CD-ROM contains a number of created variables. These created variables will be included on each public release of the NLSY97. With very few exceptions, these variables are either commonly used items that are derived from a number of different NLSY97 survey questions or longitudinal items that require updating in each round. If they were not provided, creating either the longitudinal or the cross-sectional variables might present difficulties for some NLSY97 users. In general, the created variables present information in five topical areas: education, employment, family background and formation, geographic information, and income and assets.

The first five appendices present the programs that were used to create the NLSY97 variables in round 4. The majority of programs are in SAS; a few are in SPSS. In the interest of space, four variables were not included in these appendices. These are the following:

CV_INTERVIEW_CMONTH
CV_INTERVIEW_DATE
CV_AGE_INT_DATE
CV_AGE(MONTHS)_INT_DATE

These variables are either directly picked up from the data set or relatively easy to compute. In the case of CV_INTERVIEW_CMONTH and CV_AGE(MONTHS)_INT_DATE, a simple formula of {([variable year-1980]*12)+variable month} was used to translate the dates into a continuous month scheme (see Appendix 7 for a further explanation of the continuous month scheme).

The programs in this supplement are provided for the convenience of the user and are available electronically from NLS User Services (see the Table of Contents for contact information).

Appendices 6 and 7 provide detailed information about the creation of the event history variables, a special set of variables included on the main file CD-ROM. Although programs are not included, the text of Appendix 6 describes the various status arrays available and discusses the creation and editing process. Appendix 7 explains the continuous month and continuous week systems used in the creation of event history variables and provides a crosswalk of continuous month/week and actual dates so that researchers can associate event history variables with other information about the respondents.

Appendices 10 and 11 describe the variables created from two special data collections: the administration of the computer-adaptive version of the Armed Services Vocational Aptitude Battery and the Transcript Survey, which gathered transcripts from high schools attended by selected NLSY97 respondents. Providing additional information about the educational experiences, abilities, and aptitudes of NLSY97 respondents, these data collections required specialized coding and variable creation procedures.

INDEX TO CREATED VARIABLE PROGRAMS

This section lists by question name each NLSY97 created variable for which a program is provided in this document, along with the accompanying page number for the program.

CV_AA_DEGREE, 46
CV_AMT_GOVNT_PGM_PCY.xx, 296
CV_ASSOC_CREDITS.xx, 60
CV_BA_CREDITS.xx, 60
CV_BA_DEGREE, 46
CV_BIO_CHILD_HH, 267
CV_BIO_CHILD_NR, 267
CV_CENSUS_REGION, 275
CV_CHILD_BIRTH_DATE.xx_M, _Y, 267
CV_CHILD_BIRTH_MONTH.xx, 267
CV_CHILD_DEATH_DATE.xx_M, _Y, 267
CV_CHILD_DEATH_MONTH.xx, 267
CV_CHILD_STATUS.xx, 267
CV_COHAB_TTL, 258
CV_ENROLLSTAT, 31
CV_FIRST_COHAB_DATE_M, _Y, 258
CV_FIRST_COHAB_MONTH, 258
CV_FIRST_MARRY_DATE_M, _Y, 258
CV_FIRST_MARRY_MONTH, 258
CV_GED, 46
CV_GOVNT_PGM_EVER, 296
CV_GOVNT_PGM_YR.xx, 296
CV_GRADE_SKIPPED_EVER, 48
CV_GRADE_SKIPPED_YR, 48
CV_GRADES_REPEAT_EVER, 48
CV_GRADES_REPEAT_YR, 48
CV_HGC_EVER, 31
CV_HGC_YR, 31
CV_HH_NET_WORTH, 278
CV_HH_POV_RATIO, 278
CV_HH_REL_CURRENT, 256
CV_HH_SIZE, 255
CV_HH_UNDER_18, 255
CV_HH_UNDER_6, 255
CV_HIGHEST_DEGREE_EVER, 31
CV_HIGHEST_DEGREE_YR, 31
CV_HOURS_WK_ADULT_ALL, 116
CV_HOURS_WK_ADULT_ET, 116
CV_HOURS_WK_TEEN, 116
CV_HOURS_WK_YR_ALL.xx, 110
CV_HOURS_WK_YR_ET.xx, 110
CV_HOURS_WK_YR_SE.xx, 110
CV_HRLY_COMPENSATION, 128
CV_HRLY_PAY, 128
CV_HS_DIPLOMA, 46
CV_INCOME_GROSS_YR, 278
CV_JOB<13_WKS, 128
CV_MARRIAGES_TTL, 258
CV_MARSTAT, 258
CV_MARSTAT_COLLAPSED, 258
CV_MSA, 275
CV_PIAT_PERCENTILE_SCORE, 65
CV_PIAT_STANDARD_SCORE, 65
CV_SCH_ATTEND_EVER, 57
CV_SCH_ATTEND_YR, 57
CV_SCHOOL_SIZE, 67
CV_SCHOOL_TYPE, 44
CV_STUDENT_TEACHER_RATIO, 67
CV_TRN_CERT, 69
CV_TRN_CERT_DATE~M, ~Y, 69
CV_TTL_JOB_ALL.xx, 121
CV_TTL_JOB_ET.xx, 121
CV_TTL_JOB_SE.xx, 121
CV_TTL_JOBS_ADULT_ALL, 125
CV_TTL_JOBS_ADULT_ET, 125
CV_TTL_JOBS_TEEN, 125
CV_TTL_RESIDENCES, 271
CV_URBAN_RURAL, 276
CV_WKSWK_ADULT_ALL, 99
CV_WKSWK_ADULT_ET, 99
CV_WKSWK_DLI_ALL, 98
CV_WKSWK_DLI_ET, 98
CV_WKSWK_DLI_SE, 98
CV_WKSWK_JOB_DLI.xx, 108
CV_WKSWK_JOB_YR.xx.xx, 102
CV_WKSWK_TEEN, 99
CV_WKSWK_YR_ALL.xx, 94
CV_WKSWK_YR_ET.xx, 94
CV_WKSWK_YR_SE.xx, 94
UNEMPRATE-COL, 276

NLSY97 Appendix 1:

Education Variable Creation

Enrollment Status, Highest Grade Completed, and Highest Degree Received

Variables Created: CV_ENROLLSTAT
CV_HGC_EVER
CV_HGC_YR
CV_HIGHEST_DEGREE_EVER
CV_HIGHEST_DEGREE_YR

Variables Used

Name in Program	Question Name on CD	Name in Program	Question Name on CD
sym1gr11	SYMBOL!CUR!FIRSTGRADE.01.01 (round 1)	e913131, e913132	YSCH-9131.03.01, .02
compyr2	year completed highest grade (internal variable from r2 program)	e913141, e913142	YSCH-9131.04.01, .02
		e913151	YSCH-9131.05.01
e3500r1	YSCH-3500 (round 1)	e913161	YSCH-9131.06.01
e11700r1	YSCH-11700 (round 1)	e913171	YSCH-9131.07.01
e28200r1	YSCH-28200 (round 1)	e933511M, Y	YSCH-9335.01.01~M, ~Y
t70001r1- t70003r1	YTRN-7000.01-.03 (round 1)	e933512M, Y	YSCH-9335.01.02~M, ~Y
GRSURV1	CV_HGC_EVER (round 1)	e933513M, Y	YSCH-9335.01.03~M, ~Y
DEGSURV1	CV_HIGHEST_DEGREE_EVER (round 1)	e933521M, Y	YSCH-9335.02.01~M, ~Y
e2857r2	YSCH-2857 (round 2)	e933522M, Y	YSCH-9335.02.02~M, ~Y
e3112r2	YSCH-3112 (round 2)	e933531M, Y	YSCH-9335.03.01~M, ~Y
e11700r2	YSCH-11700 (round 2)	e933541M, Y	YSCH-9335.04.01~M, ~Y
t70001r2, t70002r2	YTRN-7000.01, .02 (round 2)	e933551M, Y	YSCH-9335.05.01~M, ~Y
GRSURV2	CV_HGC_EVER (round 2)	e933561M, Y	YSCH-9335.06.01~M, ~Y
GRJUNE2	CV_HGC_YR (round 2)	e938511-e938513	YSCH-9385.01.01-.03
DEGSURV2	CV_HIGHEST_DEGREE_EVER (round 2)	e938521-e938523	YSCH-9385.02.01-.03
DEGJUNE2	CV_HIGHEST_DEGREE_YR (round 2)	e938531, e938532	YSCH-9385.03.01, .02
e1605	YSCH-1605	e938541, e938542	YSCH-9385.04.01, .02
e16151, e16152	YSCH-1615.01, .02	e938551	YSCH-9385.05.01
e16241, e16242	YSCH-1624.01, .02	e938561	YSCH-9385.06.01
e16271	YSCH-1627.01	e938571	YSCH-9385.07.01
e2806	YSCH-2806	e9589111-e9589114	YSCH-9589.01.01.01-.04
e2857	YSCH-2857	e9589121	YSCH-9589.01.02.01
e2908	YSCH-2908	e9589211-e9589213	YSCH-9589.02.01.01-.03
e3010	YSCH-3010	e9589311, e9589312	YSCH-9589.03.01.01, .02
e3061	YSCH-3061	e9589411	YSCH-9589.04.01.01
e3103, e3104	YSCH-3103, 3104	e9589511	YSCH-9589.05.01.01
e3112	YSCH-3112	e9946111-e9946114	YSCH-9946.01.01.01-.04
e3878	YSCH-3878	e9946121	YSCH-9946.01.02.01
e4793	YSCH-4793	e9946211-e9946213	YSCH-9946.02.01.01-.03
e4795	YSCH-4795	e9946311, e9946312	YSCH-9946.03.01.01, .02
e4951	YSCH-4951	e9946411	YSCH-9946.04.01.01
e54581-e54587	YSCH-5458.01-.07	e9946511	YSCH-9946.05.01.01
e67841-e67847	YSCH-6784.01-.07	e199111M, Y	YSCH-10099.01.01.01~M, ~Y
e69381-e69386	YSCH-6938.01-.06	e199112M, Y	YSCH-10099.01.01.02~M, ~Y
e694311, e694312	YSCH-6943.01.01, .02	e199113M, Y	YSCH-10099.01.01.03~M, ~Y
e694321	YSCH-6943.02.01	e199211M, Y	YSCH-10099.02.01.01~M, ~Y
e694331	YSCH-6943.03.01	e199212M, Y	YSCH-10099.02.01.02~M, ~Y
e714211	YSCH-7142.01.01	e199213M, Y	YSCH-10099.02.01.03~M, ~Y
e714221	YSCH-7142.02.01	e199311M, Y	YSCH-10099.03.01.01~M, ~Y
e714231	YSCH-7142.03.01	e11700	YSCH-11700
e714251	YSCH-7142.05.01	e11900M, Y	YSCH-11900~M, ~Y
e71921-e71927	YSCH-7192.01-.07	e13300	YSCH-13300
e841611-e841613	YSCH-8416.01.01-.03	e273371-e273375	YSCH-27337.01-.05
e841621-e841623	YSCH-8416.02.01-.03	e234501-e234503	YSCH-23450.01-.03
e841631, e841632	YSCH-8416.03.01, .02	t7000_1-t7000_4	YTRN-7000.01-.04
e841641, e841642	YSCH-8416.04.01, .02	pubid	PUBID
e841651	YSCH-8416.05.01	dths	CV_HS_DIPLOMA
e841661	YSCH-8416.06.01	dtged	CV_GED
e841671	YSCH-8416.07.01	dtaa	CV_AA_DEGREE
e913111-e913113	YSCH-9131.01.01-.03	dtba	CV_BA_DEGREE
e913121-e913123	YSCH-9131.02.01-.03	roscode1-roscode5	NEWSCHOOL_SCHCODE.01-.05

Codes for Created Variables

Enrollment Status (CV_ENROLLSTAT)

1. not enrolled, no high school degree, no GED
2. not enrolled, GED, but no high school degree
3. not enrolled, a high school degree, no GED
4. not enrolled, some college
5. not enrolled, 2 year college graduate
6. not enrolled, 4 year college graduate
7. not enrolled, graduate degree
8. enrolled in grades 1-12
9. enrolled in a 2 year college
10. enrolled in a 4 year program
11. enrolled in a graduate program

Highest Grade Completed (CV_HGC_EVER, CV_HGC_YR)

0. none
1. first
2. second
3. third
4. fourth
5. fifth
6. sixth
7. seventh
8. eighth
9. ninth
10. tenth
11. eleventh
12. twelvth
13. first year college
14. second year college
15. third year college
16. fourth year college
17. fifth year college
18. sixth year college
19. seventh year college
20. eighth year college
95. ungraded

Highest Degree Received (CV_HIGHEST_DEGREE_EVER, CV_HIGHEST_DEGREE_YR)

0. none
1. GED
2. high school diploma
3. junior college or 2-year associate degree
4. bachelor's degree (B.A., B.S., or unspecified)
5. master's degree
6. doctoral degree
7. professional degree (M.D., L.L.D., D.D.S., J.D.)

This program first creates an enrollment status variable for each respondent. It then identifies the highest grade completed by the respondent as of the round 4 interview date and as of June 30, 2000. Finally, it determines the highest degree or diploma received by the respondent as of the same two dates.

Notes on education variables: Respondents who report receiving both a GED and a high school diploma are coded as receiving a high school diploma. Respondents who report completing 12th grade but do not report receiving a diploma or GED are coded as having completed 12th grade (in the GRSURV and GRJUNE variables, if applicable) but are not credited with a diploma/GED in the ENCAT, DEGSURV, or DEGJUNE variables. Researchers interested in diploma receipt should use the DEGSURV or ENCAT variables and should not assume that completion of 12th grade signifies receipt of a high school diploma.

Some respondents provided conflicting information about grades attended/completed in the schooling loops (e8416 and e9589) versus the single items (e3112, e2857). In Round 4, approximately 148 respondents have conflicting grade reports, identified using the code:

```
where (enroll=0 and 12>=e3112^=grsurv>0 and degsurv not in (1,2) and grsurv^=95)
or (0<maxgrade=grsurv<95 and grsurv^=e3112 and e3112<=12)
```

This program gives priority to information provided in the schooling loops over that provided on single item questions. Users may wish to check the CV_HGC_EVER variable against student single item self-reports of HGC (e3112), which in many cases indicates a higher grade was completed than that reported in the loops.

Thirty-six respondents gave irreconcilable conflicting reports regarding their school attendance in the loops versus the single-item sections of the questionnaire. These respondents are assigned invalid skips (-3) because no reasonable determination of their enrollment status and grades completed can be made. Specifically, respondents are assigned a –3 if they a) report being not enrolled, b) report a highest grade completed on the single items (e3112) that is greater than the CV_HIGHEST_DEGREE_EVER for the prior round, but do not indicate completing this grade in a schooling loop, and c) report that their highest grade attended (e2857) is two or more greater than the highest grade completed in the previous round. If these respondents provide valid information on degrees earned, that variable will have a valid value.

```
/*************** ENROLLMENT STATUS **************/*
Initialize variables;
enroll=-3;          *Indicator of whether respondent currently enrolled ;
encat=-3;           *Enrollment status variable (including educational attainment) ;
dip=0;              *Indicator of whether respondent has a hs diploma (check previous rounds);
ged=0;              *Indicator of whether respondent has a GED (check previous rounds);
hs=0;               *Indicator of whether respondent is currently attending grades 1 to 12 ;
college=0;          *Indicator of whether respondent is current attending 2 or 4 year college ;
flagcol=0;          *Indicator to flag respondents currently attending grades 1 to 12 and enrolled in college ;
flagloop=0;         *Indicator to flag respondents going through old school re-enrollment loop twice ;

* Defining arrays for use later in the program;
array e5458 e54581 e54582 e54583 e54584 ;
array e6784 e67841 e67842 e67843 e67844 ;
array e7192 e71921 e71922 e71923 e71924 ;
array e23450 e234501 e234502 e234503;
array e27337 e273371 e273372 e273373 e273374;
array t7000a t7000_1 t7000_2 t7000_3 t7000_4 t70001r1 t70002r1 t70003r1 t70001r2 t70002r2 t70001r3 t70002r3
        t70003r3 t70004r3;
array e6938 e69381 e69382 e69383;
array e69431 e694311 e694321 blah;          /*"Blah" is a filler variable to make the array match in N*/
array e7142 e714211 e714221 e714231;
array roscode roscode1 roscode2 roscode3 roscode4 roscode5;

somecoll=0; /*flag to indicate whether R attended a college (in a degree program) in current or previous rounds*/

**HAND EDIT Patch to fix problems on GED receipt from R2;
if id=1117361 then do; ged=1; dtged= 208;  end;
if id=1214251 then do; ged=1; dtged= 209;  end;
if id=1223531 then do; ged=1; dtged= 221;  end;
if id=1371591 then do; ged=1; dtged= 223;  end;
if id=1417231 then do; ged=1; dtged= 227;  end;
if id=1455241 then do; ged=1; dtged= 219;  end;
if id=1508031 then do; ged=1; dtged= 217;  end;
if id=1508291 then do; ged=1; dtged= 207;  end;
if id=1787051 then do; ged=1; dtged= 222;  end;
if id=1703251 then ged=1; /*reports receipt of GED in comments on e13900*/

*ENROLLMENT STATUS Begin by determining whether the respondent is enrolled;
   *Note that summer vacations are included as part of the enrollment period;
     if e4951=-5 then do; enroll=-5; encat=-5; end;
* Continuously enrolled in old school since dli (or start date);
     if e4795=1 or e4793=1 then enroll=1;
* Not continuously enrolled and not enrolled in any other schools since;
     if e4795=0 and e4951=0 then enroll=0;
* Not continuously enrolled in old school since dli, not re-enrolled in old school since dli, and no new schools since dli;
     if e1605=-2 then enroll=-2;
     if e4795=0 and e1605=0 and e4951=0 then enroll=0;
     if e4793=0 and e1605=0 and e4951=0 then enroll=0;
     if e3878=0 then enroll=0;
* Not continuously enrolled in old school since dli, have reenrolled in old school a maximum of 1 time, have not reported
        any other schools attended since;
     if e4793=0 and e1605=1 and e16151=1 then enroll=1;
* Not continuously enrolled in old school since dli, have received degree and/or completed course work at old school (i.e.,
        YSCH-1600=1), and has not reported new school since;
     if e4793=0 and e4951=0 and e1605=-4 then enroll=0;
```

```
*RE-ENROLLMENT IN OLD SCHOOL  Note that respondents can only re-enroll loop once and it is not possible to
        determine which individuals re-enrolled additional times  Individuals who were not continuously enrolled up to
        the interview during the last enrollment spell and who do not report any new schools will be coded as not
        enrolled;
  *Flag those who re-enroll;
     if e16241=1 then flagloop=1;
     if (e4795=0 and e1605=1) then do;
          if e16151=1 then enroll=1;                          * continuously enrolled in old school;
          if e16151=0 and e16241=0 and e16271=0 and e4951=0 then enroll=0;
          * not continuously enrolled in old school, first time through loop,
          no other re-enrollments at old school and no new schools reported ;
     end;

* ENROLLMENT IN NEW SCHOOLS;
     if (e4951=1 or e3878=1) then do;
          do i=1 to 4;                                        * allow for up to four new schools to be recorded;
               if e6784(i)=1 and e7192(i)=0 then enroll=1;    * continuously enrolled in new school, no other new
                                                              schools reported;
               if e6784(i)=0 and e7192(i)=0 then enroll=0;    * not continuously enrolled in new school, did not re-
                                                              enroll in new school, no other new schools reported;
          end;
      end;

* make sure that all members of the e6784 array are in the dataset - R's who leave a new school because they graduated or
        finished coursework will not be asked this question;

* RE-ENROLLMENT PERIODS IN NEW SCHOOLS;
     if (e4951=1 or e3878=1) then do;
          do i=1 to 3;                                        * allow for up to three times through this loop;
               if (e6784(i)=0 and e6938(i)=1) then do;
                    if e69431(i)=1 and e7192(i)=0 then enroll=1; * continuously re-enrolled in new school, no other new
                                                              schools reported;
                    if e69431(i)=0 and e7142(i)=0 and e7192(i)=0 then enroll=0; * not continuously re-enrolled in new
                                                              school, not re-enrolled in new school for a subsequent
                                                              spell, no other new schools reported ;
               end;
          end;
     end;

* ENROLLMENT STATUS CATEGORIES;
*First create variables to flag those who have ever received a GED or hs diploma;
* DIP is set to 1 if R received HS diploma in rounds 1 through 4;
     if e11700=1 or e11700r1=1 or e26700r1=1 or e11700r2=1 or e11700r3=1 then dip=1;
     else if e11700 in (-1, -2, -3) or e11700r1 in (-1, -2, -3) or e26700r1 in (-1, -2, -3) or e11700r2 in (-1, -2, -3) or
        e11700r3 in (-1, -2, -3) then dip=-3;

* GED is set to 1 if R received GED in rounds 1 through 4;
* NOTE:  e13300 is not available for R2--use CV (encat2) from R2;
     if e28200r1=1 or e13300=1 or e13300r3=1 or e234501=9 or e234502=9 or encat2=2 then ged=1;
     else if e28200r1 in (-1, -2, -3) or e13300 in (-1, -2, -3) then ged=-3;
     do i=1 to 13;
       if t7000a(i)=4 then ged=1;
       if ged=0 and t7000a(i) in (-1, -2, -3) then ged=-3;
     end;

* NOT ENROLLED;
if (enroll=0) then do;        * Flag to indicate that R attended college (in a degree program) in a previous round;
```

```
    if encat3 in (4,5,6,7,9,10,11) and degsurv3 in (1,2) then somecoll=1;
    if (collr2=1) then somecoll=1;
    if encat1 in (4,5,6,7,9,10,11) and degsurv1 in (1,2) then somecoll=1;

    * No high school degree, no GED;
    if ged le 0 and dip le 0 then encat=1;

    * GED;
    if ged=1 then encat=2;

    * High school degree;
    if dip=1 then encat=3;

    * Highest grade attended is at least 1 year of college;
    * R reports a college on the roster and indicates being enrolled in a college degree program (e27337<7 and
        e27337^=2) and R has HS diploma/GED;
    * PLEASE NOTE:  Those who indicated attending a college and working toward a "vocational or technical
        certificate" (e27337=2) are coded as not having attended college (because these credits typically won't transfer
        for a BA or AA degree;
    do i=1 to 4;
      if roscode(i) in (4,5) and e27337(i) in (-2,-3) then do; encat=-3; end;
      if dip=1 and roscode(i) in (4,5) and (e27337(i)>6 or e27337(i)=2) and somecoll=0 then do; encat=3; end;
      if ged=1 and dip=0 and roscode(i) in (4,5) and (e27337(i)>6 or e27337(i)=2) and 13<=e2857<=20 and
        somecoll=0 then do; encat=2; end;
      if ged=0 & dip=0 and roscode (i) in (4,5) and (e27337(i)<=6 or e27337(i)=2) and 13<=e2857<=20 then encat=-3;
    end;

/* if R attended any college (and was enrolled in a degree program), but is not currenly enrolled encat=4 since they
        attended some college;*/
    prevrnd=0;
    do i=1 to 4;
      if roscode(i) in (4,5) and (1<=e27337(i)<=6 and e27337(i)^=2) and (13<=e2857<=20 or e3112>=12) then
        encat=4; /*r attended some college in current round*/
    end;

    if somecoll=1 then encat=4; /*if R attended college in a prior round*/
    * Received junior college or 2 year associate degree (need to check each school);
    if (e234501=1 or e234502=1 or e234503=1) then encat=5;
    * Received bachelor's degree (need to check each school);
    if (e234501=3 or e234502=3 or e234503=3) then encat=6;
    * Received master's, doctoral or professional degree (need to check each school) ;
    do i=1 to 3; if e23450(i) in(4,5,6) then encat=7; end;

end;

flagvoc=0; /*flag for Rs enrolled in "vocational/technical certificate program"*/
do i=1 to 4; if e27337(i)=2 then flagvoc=1; end;
nodegree=0; /*flag for Rs enrolled in non-degree college programs*/

* ENROLLED ;
if (enroll=1) then do;                    * missing values ;
    if (roscode1=-2 or roscode2=-2 or roscode3=-2 or roscode4=-2) then encat=-3;
    if (e273371=-2 or e273372=-2 or e273373=-2)then encat=-3;
    if (e273371=-2 or e273372=-2 or e273373=-2)then college=-3;
    if (e273371=-2 or e273372=-2 or e273373=-2)then flagcol=-3;

    * Highest grade attended as of today less than or equal to 12th grade and don't have a diploma/GED ;
```

```
    if 1<=e2857<=12 and dip le 0 and ged le 0 then encat=8;

**** Use roster info on sch enrollment to identify type of school enrolled in. 2year/4year college should refer to the
        TYPE OF INSTITUTION (from roster info) not the TYPE OF DEGREE PROGRAM (since many attending
        2-year schools report being in a 4-year degree program because that is their ultimate degree goal, thus the type of
        institution better reflects their current enrollment status);
*Those who completed high school (or a GED), but report being enrolled in a grade>12 and do not indentify a college on
        the school rosters are coded as -3 on encat;
    do i=1 to 4;
      if enroll=1 and (dip=1 or ged=1) and 95>e2857>=13 and roscode(i) in (1,2,3,6) then encat=-3;
     end;

*Those who completed high school (or a GED), but report being enrolled in a grade<=12 are coded as -3 on encat;
     do i=1 to 4;
      if enroll=1 and (dip=1 or ged=1) and e2857<=12 then encat=-3;
    end;

*if enr. in 2-year school, working toward a degree, and current grade is greater than high school then encat=2 year college;
*Only use up to roscode4 since there are no college attendees in roscode6 or 7;
*if R enr. in a 2/4 year school (roscode in 4,5) and e27337=9 and R doesn't have a diploma then R is working toward a
        GED;
    do i=1 to 4;
      if dip=1 and roscode(i)=4 and e27337(i)=2 then do; encat=3; nodegree=1; end;
      if ged=1 and roscode(i)=4 and e27337(i)=2 then do; encat=2; nodegree=1; end;
      if ged=0 and dip=0 and roscode(i) in (4,5) and e27337(i)=9 then encat=8;
      if ((roscode(i)=4 and e27337(i) in (1,3,4,5,6)) and e2857 ge 13 and e2857 lt 95) then encat=9;
    end;

*if enr. in 4-year school, working toward a degree and current grade is greater than high school then encat=4 year college;
    do i=1 to 4;
      if dip=1 and roscode(i)=5 and e27337(i)=2 then do; encat=3; nodegree=1; end;
      if ged=1 and roscode(i)=5 and e27337(i)=2 then do; encat=2; nodegree=1; end;
      if ((roscode(i)=5 and e27337(i) in (1,3,4,5,6)) and e2857 ge 13 and e2857 lt 95) then encat=10;
    end;

/*Rs who have obtained a BA, are still enrolled in a post-secondary institution, and report working toward a graduate or
        professional degree should be coded as enrolled in a graduate program (this code is based on degree program,
        not type of institution--again, there is a reporting problem here, with some whose ultimate degree goal is a post-
        baccalaureate degree indicating that they are currently working toward the higher degree, even though they have
        not yet completed a BA/BS*/
*Enrolled in a university, working towards a graduate degree and current grade is greater than or equal to 4th year college;
    do i=1 to 4;
      if roscode(i)=5 and (4<=e27337(i)<=6) and 16<=e2857 lt 95 then encat=11;
    end;
end;

* The following is for respondents who are enrolled, have no evidence of a high school diploma, yet reported that highest
        grade attended was 1st year of college. Since there is no evidence these people have a diploma and have not
        attended a new school, they will be placed under enrolled in grades 1-12 ;
    flagged=0;
    if enroll=1 and e2857=13 and dip=0 and ged=0 then encat=8;
    do i=1 to 4;
      if enroll=1 and roscode(i) in (4.5) and e2857>=13 and dip=0 and ged=0 and e27337(i) in (7,8) then encat=8;
      if enroll=1 and roscode(i) in (4,5) and e2857>=13 and dip=0 and ged=0 and e27337(i) in (7,8) then do;
        flagged=1; nodegree=1; end;
    end;
```

```
* The following is for respondents who have a high school diploma, a valid date for the diploma, have not enrolled in a
        new school since dli, and report the highest grade completed being 12.  They have been categorized as not
        enrolled with a high school diploma (encat=3) despite the fact that these respondents claim to be continuously
        enrolled since dli.  Since there is no evidence of college from the respondent, they have been grouped as not
        enrolled;
    if e11700=1 and e11900m>0 and e11900y>0 and e4951=0 and e3112=12 then encat=3;

* The following is for respondents who report the highest grade attended as ungraded (YSCH-2857=95), show no
        evidence of a high school diploma, were skipped out of the homeschool question, and are enrolled.  Again, since
        there is no evidence of a high school diploma, they are regarded as enrolled between grades 1 and 12 ;
    if e2857=95 and dip=0 and ged=0 and enroll=1 then encat=8;

*The following is for Rs who received a HS diploma, provide a valid date for diploma receipt, yet also report current
        enrollment in a high school.  Because these Rs have obtained a diploma, they are characterized as 3, not enrolled,
        a high school degree, no GED;
    flagende=0; /*if r reports being enrolled and already has a diploma*/
    do i=1 to 5;
     if enroll=1 and e11700=1 and e11900m>0 and e11900y>0 and e3112=12 and roscode(i)=3 then do; encat=3;
        flagende=1; end;
    end;

* Flag individuals who are enrolled in grades 1 to 12 and are enrolled in 2 or 4 year colleges (could be getting credits
        towards GED)  For created enrollment status variable above, the flagged individuals will fall into category 8 ;
    if enroll=1 and 1<=e2857<=12 and dip=0 and ged=0 then hs=1;
    if enroll=1 and (e273371=1 or e273372=1 or e273373=1 or e273374=1 or e273371=3 or e273372=3 or e273373=3
        or e273374=3) then college=1;
    if hs=1 and college=1 then do; flagcol=1; encat=8; end;

*Those who completed high school, report being enrolled in college, but are in a non-degree program are coded as not
        enrolled, a high school degree (3).  If they previously attended a college in a degree program they are coded as
        not enrolled, some college (4).  Those whose most recent enrollment is in a degree program are coded as enrolled
        in a 2year (encat=9) or 4year (encat=10) school;
    * Flag to indicate that R attended college (in a degree program) in a previous round (who did not correct HGA as of
        last int. (e2806) to indicate that HGA <13;
    if encat3 in (4,5,6,7,9,10,11) and ((e2806 in (-4,-5)) or (e2806>=13)) then somecoll=1;
    if (collr2=1) then somecoll=1;
    if encat1 in (4,5,6,7,9,10,11) and degsurv1 in (1,2) then somecoll=1;
    do i=1 to 4;
      if enroll=1 and roscode(i) in (4,5) and e27337(i) in (1,3,4,5,6) then somecoll=1;
      if enroll=1 and dip=1 and roscode(i) in (4,5) and e27337(i) in (2,7,8) and somecoll=0 then do; encat=3;
        nodegree=1; end;
      if enroll=1 and ged=1 and roscode(i) in (4,5) and e27337(i) in (2,7,8) and somecoll=0 then do; encat=2;
        nodegree=1; end;
      if enroll=1  and roscode(i) in (4,5) and e27337(i) in (2,7,8) and somecoll=1 then encat=4;
      if enroll=1  and roscode(i) in (4,5) and e27337(i) in (2,7,8) and somecoll=1 then encat=4;
      if enroll=1  and roscode(i)=4 and e27337(i) in (1,3,4,5,6) then encat=9;
      if enroll=1  and roscode(i)=5 and e27337(i) in (1,3,4,5,6) then encat=10;
    end;

check=0;
if e4795=1 and encat in (1,2,3,4,5,6,7) and flagende=0 and nodegree=0 then check=1;

**Rs who report continuous enrollment in a DLI school, who subsequently report an additional school (typically the first
        school is a 4 year college and R is working toward a BA and the second school is a 2/4 year college R attends in
        the summer in a non-degree program.  This code overwrites code above that erroneously replaces the continuous
        enrollment information with the more recently reported non-degree school.  A report of continuous enrollment
```

```
        in high school is overwritten with college enrollment if the R reports earning a diploma or GED--Continuous
        enrollment in a high school is overwritten with encat in (2,3) depending on degree reported;
**Must fix "ENROLL" flag because this flag is used in the grades skipped/repeated program;
**NOTE:  if ROSCODE1=2 and ROSCODE2-4 in (4,5) then the case must be hand edited to ensure accuracy--there is
        only one such case in R4;
if e4795=1 and roscode1=4 and e273371 in (1,3,4,5,6) then do; encat=9; enroll=1; end;
if e4795=1 and roscode1=5 and e273371 in (1,3,4,5,6) then do; encat=10; enroll=1; end;
if e4795=1 and roscode1 in (2,3) and dip=0 and ged=0 then do; encat=8; enroll=1; end;
if e4795=1 and roscode1=2 and roscode2=3 and dip=0 and ged=0 then do; encat=8; enroll=1; end;

do i=2 to 4;
  if e4795=1 and roscode1=3  and roscode(i)=4 and e27337(i) in (1,3,4,5,6) then do; encat=9; enroll=1; end;
  if e4795=1 and roscode1=3 and roscode(i)=5 and e27337(i) in (1,3,4,5,6) then do; encat=10; enroll=1; end;
  if e4795=1 and roscode1=3 and roscode2 in (-4,1,2,3) and roscode3 in (-4,1,2,3) and roscode4 in (-4,1,2,3) then do;
    if dip=1 then encat=3;
    if ged=1 then encat=2;
  end;
end;

flagndeg=0;
do i=1 to 4; if e27337(i) in (2,7,8) then flagndeg=1; end;

/***********HAND EDITS***********/
/*the cases below were found using the code: "where dip=0 and ged=0 and encat in (4,5,6,7,9,10,11);"*/
if id=1086801 then encat=-3;                    /*invalid code for type of college program in e27337*/
if id=1392801 then encat=8;                     /*enr. in both HS and College, not done with HS*/
if id=1296591 then encat=8;                     /*same as above*/
if id=1693361 then do; encat=2; ged=1; end;  /*r indicates GED receipt on comments in YSCH-13900*/
if id=1068321 then encat=8;                     /*comments on YSCH-13900*/
if id=1100031 then encat=8;                     /*comments on YSCH-13900*/
if id=1253151 then ged=1;                       /*comments on YSCH-13900*/
if id=1253411 then ged=1;                       /*comments on YSCH-13900*/
if id=1253681 then encat=8;                     /*concurrent enrollment in HS and College-YSCH-13900*/
if id=1289541 then encat=8;                     /*concurrent enrollment in HS and College*/
if id=1407971 then encat=8;                     /*concurrent enrollment in HS and College*/
if id=1470422 then encat=8;                     /*concurrent enrollment in HS and College*/
if id=1473461 then encat=8;                     /*concurrent enrollment*/
if id=1477622 then encat=8;                     /*concurrent enrollment*/
if id=1536031 then encat=8;                     /*completing HS diploma in a college*/
if id=1549561 then encat=8;                     /*concurrent enrollment*/
if id=1552721 then encat=8;                     /*concurrent enrollment*/
if id=1571791 then encat=8;                     /*working on GED at a junior college*/
if id=1630362 then encat=8;                     /*concurrent enrollment*/
if id=1695621 then encat=8;                     /*concurrent enrollment*/
if id=1740801 then encat=8;                     /*working on GED at a junior college*/
if id=1789261 then encat=8;                     /*enrolled in "adult high school" program at a college*/
if id=1929712 then encat=8;                     /*concurrent enrollment*/
if id= 1162781 then encat=10; /*R completed 12th grade and then started attending college, school code incorrect*/
```

/******** HIGHEST GRADE COMPLETED AS OF THE SURVEY DATE (GRSURV)**********/**

```
* Create four arrays: the first array includes items that ask if R has completed for which at least one R answered positively,
        the second and third arrays list all of the questions giving the month and year of completion, and the fourth
        array lists the questions indicating the grades that the R may or may not have completed.  ROSCODE and
        e27337 arrays are also employed to ensure consistency between ENCAT and GRSURV variables--if R reports
```

```
        attending a grade that is > 12 but doesn't report being enrolled in a degree program at a college or university,
        they are coded as having completed 12th grade;
array logic e913111 e913112 e913121 e913122 e913131 e913141 e913151 e9946111 e9946112 e9946113 e9946114
        e9946211 e9946212 e9946311 e9946312;
array cyear e933511y e933512y e933521y e933522y e933531y e933541y e933551y e199111y e199112y e199113y
        e199114y e199211y e199212y e199311y e199312y;
array cmonth e933511m e933512m e933521m e933522m e933531m e933541m e933551m e199111m e199112m
        e199113m e199114m e199211m e199212m e199311m e199312m;
array grade e841611 e841612 e841621 e841622 e841631 e841641 e841651 e9589111 e9589112 e9589113 e9589114
        e9589211 e9589212 e9589311 e9589312;

maxgrade=-16; compmon=-16; compyr=-16; everloop=0; badgrade=0;
assign='unassigned';

**Select cases where there is a valid grade and it is greater than the previously recorded grade;
do i=1 to 15;
  if grade(i) ge 0 and grade(i) gt maxgrade then do;
    if logic(i)=1 then do; assign='from loop'; maxgrade=grade(i); compmon=cmonth(i); compyr=cyear(i);
      if cyear(i) lt 2000 or (cyear(i) eq 2000 and cmonth(i) le 6) then do; grjune=grade(i); cjumon=cmonth(i);
        cjuyr=cyear(i); end;
    end;
    else if logic(i) in (-1, -2, -3) then do; maxgrade=logic(i); assign='logic missing  '; end;
  end;
  if grade(i) in (0,-1, -2, -3,-4) and logic(i) gt -4 then do; maxgrade=-55; assign='grade invalid  '; end;
  if logic(i) gt -4 then everloop=1;    *Flag all observations that go through a grade loop at least once;
end;

*Create variable indicating highest grade attended from loops;
hgaloop=e841611;
do i=2 to 15;
   if grade(i) gt hgaloop then do; hgaloop=grade(i); end;
end;

*Create variable indicating grsurv for previous interview;
grsprev=grsurv3;
if grsurv3 lt 0 then grsprev=grsurv2;
if grsurv2 lt 0 then grsprev=grsurv1;
if e3061 ge 0 then grsprev=e3061;

*Create final GRSURV variable;  ****Make GRSURV equal to maximum grade;
grsurv=maxgrade;
**** If non-interview then grsurv=-5;
if e3112=-5 then do; grsurv=-5; assign = 'non-interview'; ged=-5; dip=-5;
end;

**If invalid or incorrect grade data was read into loop or if r. never goes through the loops (usually because they now
        attend 2 or 4 year colleges, sometimes because of infosheet error) AND they attended school at some time either
        at dli or since dli, then replace grsurv with single item.  HOWEVER:  this single item must be checked against
        the roster data:  if R completed high school, never reports attending a college in the loop/roster information, but
        indicates enrollment in a grade > 12 in the single item measure, the GRSURV variable is set to 12 since no
        college degree program has been reported;
subsitem=0;
if ((e841611 le 0 and e913111 ne -4) or grsurv=-55 or everloop=0) and e3878 ne 0 and e3112 ne -5 then do;
  do i=1 to 4;
    if e3112>12 and roscode(i) in (4,5) and e27337(i) in (2,7,8,9) then grsurv=12;
    if e3112>12 and encat in (4,5,6,7,9,10,11) then grsurv=e3112;
    if e3112>12 and (dip=0 and ged=0) and roscode(i) ne 4 and roscode(i) ne 5 then grsurv=-16;
```

```
    if e3112>=12 and roscode(i) in (4,5) and e27337(i) in (1,3,4,5,6) then grsurv=e3112;
    if e3112<12 and (dip=0 and ged=0) then grsurv=e3112;
    if e3112>=12 and dip=0 and ged=0 and roscode(i) in (1,3,6) then grsurv=-16;
    subsitem=1;
    if e841611 le 0 and e913111 ne -4 then assign= 'Enrolled, bad grade:  subs. item';
    else if everloop=0 then assign='Enrolled, not thru loops: subs. item';
    else if maxgrade=-55 then assign='Enrolled, bad grade: subs. item';
  end;
end;

***If r. was enrolled but didn't confirm completing a grade, we substitute a value equal to one less than highest grade
        attended from loop using a conservative approach:
      * if the hga from loop is equal to the single item hga (e2857) substitute hgaloop-1
      * otherwise if the hga from loop minus 1 is equal to the single item hgc (e3112) substitute hgaloop-1
      * otherwise if the the single item hgc (e3112) is equal to the previous hgc then substitute e3112
      * if none of the other statements are true and hgaloop-1 is greater than e3112 substitute e3112;
subshgam=0;
tempflag=0;
if grsurv eq -16 and enroll=1 then do;
  if hgaloop=e2857 then do; subshgam=1; grsurv=hgaloop-1; assign= 'Enrolled-no loop hgc-hgaloop eq 2857: subs
        hgaloop-1'; end;
  else if hgaloop-1 eq e3112 then do; subshgam=1; grsurv=hgaloop-1; assign= 'Enrolled-no loop hgc-hgaloop-1 eq e3112:
        subs hgaloop-1'; end;
  else if grsprev eq e3112 then do; subsitem=1; grsurv=e3112; assign= 'Enrolled-no loop hgc-e3112 eq grsprev: subs item';
        end;
  else if hgaloop-1 gt e3112 then do; subsitem=1; grsurv=e3112; assign= 'Enrolled-no loop hgc-hgaloop-1 gt item: subs
        item'; end;
  else if hgaloop le e3112 and grsprev lt e3112 then do; subsprev=1; grsurv=grsprev; assign= 'Enrolled-no loop hgc-said
        no to grade: subs prev'; end;
end;

***If someone confirmed completing a grade in the loop and didn't know the year of completion, then use the loop data
        or the single item (YSCH-3112) whichever is lower;
dateloop=0;
dateitem=0;
if maxgrade=grsurv and compyr=-2 and maxgrade gt e3112 then do; dateitem=1; grsurv=e3112; assign= 'subs. item: no
        date item lower'; subsitem=1; end;
else if maxgrade=grsurv and compyr=-2 and maxgrade le e3112 then do; dateloop=1; grsurv=maxgrade; assign= 'subs.
        loop: no date loop data lower'; subsitem=1; end;
else if compyr ne -2 then do; dateloop=.; dateitem=.; end;

***If someone reported receiving a ged or h.s. diploma, then set their hgc to 12;
set12=0;
if (dip=1 or ged=1) and grsurv lt 12 and e3112 ne -5 then do; set12=1; grsurv=12; assign= 'set to 12: dip or ged'; end;

**College attendance is not reported in the loops and college year completion should therefore make use of the single item
        HGC, but only if enrollment in a college degree program (AA, BA, MA, etc) is reported (E27337) in the current
        or a previous round.  Use ENCAT to determine whether R was ever in college.  BUT if e3112<12 and R reports
        completing a High School diploma or GED then GRSURV is set equal to 12;
if encat in (4,5,6,7,9,10,11) and 95>=e3112>=12 then do; grsurv=e3112; assign='college'; end;
if encat in (4,5,6,7,9,10,11) and 95=>e3112<12 and (dip=1 or ged=1) then do; grsurv=12; set12=1; assign='college, set to
        12'; end;
if encat in (4,5,6,7,9,10,11) and 95>e3112<12 and grsurv3>12 and encat3 in (4,5,6,7,9,10,11) then do;
  if e3061<0 then grsurv=grsurv3;
  if e3061>0 then grsurv=e3061;
  assign='college, previous round';
end;
```

```
if (encat in (4,5,6,7,9,10,11) and 95>=e3112<12) or (encat in (4,5,6,7,9,10,11) and e3112<0) then do; grsurv=-3;
        assign='college, unclear'; end;

****If R reports completing a grade in a previous round that is > GRSURV for the current round and the single item
        provides a more consistent grade, the single item is used.  If the single item is also inconsistent, the GRSURV
        value from the previous round is used;
disagree=0;
oldgrade=grsurv3;  if grsurv3<0 then oldgrade=grsprev;
if e3061>0 then oldgrade=e3061; /*use corrections of previous round info*/
if 0<oldgrade<95 and oldgrade>GRSURV>0 then disagree=1;
if disagree=1 then do;
  if (95>e3112>12 or 95>oldgrade>12) and encat in (4,5,6,7,9,10,11) then do; /*don't use single item of >12 if no
        college reported*/
    if oldgrade=(e3112-1) then grsurv=e3112;
    if oldgrade=(e3112-2) then grsurv=e3112;
    if oldgrade=e3112 then grsurv=e3112;
    if oldgrade>e3112 then grsurv=oldgrade;
    assign='disagree:  college reported, single item or prior round';
  end;
  if e3112<=12 then do;
    if oldgrade=(e3112-1) then grsurv=e3112;
    if oldgrade=(e3112-2) then grsurv=e3112;
    if oldgrade=95 then grsurv=e3112;
    if oldgrade=e3112 then grsurv=e3112;
    if oldgrade>e3112 then grsurv=oldgrade;
    assign='disagree:  replace with single item or prior round';
  end;
  if (95>e3112>12 or 95>oldgrade>12) and encat in (1,2,3,8,-3) then do; grsurv=-3; assign='disagree: no college
        reported'; end;
end;

**** Not enrolled: Assign previous highest grade completed and create a variable flagging observations that substituted the
        value of the previous year - if previous HGC is missing, then assign the single item;
subsprev=0;
if grsurv in (-16,-3) and enroll=0 and grsprev ge 0 then do;
   if compyr3 gt 0 and grsurv3 ge 0 then do;
      if e3061<0 then grsurv=grsurv3;
      if e3061>0 then grsurv=e3061;
      subsprev=1;
      assign= 'Not enrolled-r3 value from loop: subs. r3';
   end;
   else if compyr3 le 0 then do; grsurv=e3112; subsitem=1; assign= 'Not enrolled-r3 value not from loop: subs. item';
        end;
end;
else if grsurv in (-16,-3) and e3878=0 and grsprev lt 0 then do; grsurv=e3112; subsitem=1; assign= 'subs. item: not
        enrolled,bad data prev'; end;

****Flag Rs whose response highest grade completed question doesn't agree with their highest grade completed obtained
        from the grade completion loops;
diffhgc=0;
if maxgrade ne e3112 and e3112 ge 0 and maxgrade ge 0 then diffhgc=1;
if maxgrade lt 0 or e3112 lt 0 then diffhgc=.;

**Hand edit case with inconsistent data on CVs**;
if id=1253151 then grsurv=-3; /*r gives valid date for AA degree receipt but says HGC=11*/

if grsurv=0 then grsurv=-3;
```

```
if grsurv in (-55,-2,-3,-16) then grsurv=-3;


/*************** HIGHEST GRADE COMPLETED AS OF JUNE 30, 2000 (GRJUNE)**************/
/*calculate date in continuous month of receipt of ged or hs diploma*/
/*Those who completed a grade that is > 12 do not provide completion dates for each college year (only for degrees).
Thus, all Rs who attended college are coded as -4 on this variable since we do not have complete grade completion data*/
prev=0;
if grjune = . then do;
  if e3112 = -5 then grjune=-5;
  else if set12 = 1 then grjune=-3;
  else if (subsitem=1 or subshgam=1) and (grsurv le 12 or grsurv=95) then grjune=-3;
  else if (subsitem=1 or subshgam=1) and grsurv gt 12 and grsurv lt 95 then grjune=-4;
  else if e3878=0 or subsprev=1 then grjune=grsurv;
  else if assign= 'Hand edit because HGC is greater than HGA' and grsurv=grsurv3 then grjune=grsurv;
  else if assign= 'Hand edit because HGC is greater than HGA' and grsurv ne grsurv3 then grjune=-3;
  else if assign='Enrolled-grade not confirmed in loop-r2 loop good:subs r2' then grjune=grsurv;
  if set12=1 and ged=1 then do;
    if dtged gt 0 and dtged le 246 then grjune=grsurv;
    else if dtged gt 246 and maxgrade gt 0 then grjune=grsurv;
    else if dtged gt 246 and maxgrade le 0 and subsprev=1 then grjune=grsprev;
    else if dtged gt 246 and maxgrade le 0 and subsitem=1 then grjune=e3112;
  end;
  if set12=1 and dip=1 then do;
    if dths gt 0 and dths le 246 then grjune=grsurv;
    else if dths gt 246 and maxgrade gt 0 then grjune=grsurv;
    else if dths gt 246 and maxgrade le 0 then grjune=grsprev;
    else if dths gt 246 and maxgrade le 0 and subsitem=1 then grjune=e3112;
  end;
  if 0<dths le 246 and grsurv=12 then grjune=12;
  if 0<dtged le 246 and grsurv=12 then grjune=12;
  if 0<dths le 246 and grsurv>12 then grjune=-4;
  if 0<dtged le 246 and grsurv>12 then grjune=-4;
  if enroll=0 and e3010=0 and maxgrade<0 and grsurv=e3061 then grjune=e3061;
  if grsurv=grsprev then do; grjune=grsurv; if enroll=1 then funny=1; end;
  if 0<grsurv=grsurv3=e3061 and grjune<grsurv then grjune=grsurv;
    if compyr gt 0 and maxgrade=grsurv then do;
      if compyr in (1997,1998,1999) or (compyr=2000 and compmon ge 1 and compmon le 6) then grjune=grsurv;
      else if (compyr eq 2000 and compmon gt 6) or compyr gt 2000 then do; grjune=grsprev; prev=1; end;
    end;
  end;
  else if compyr=-2 then grjune=-3;
  if grjune=. or grjune=0 then grjune=-3;

/*if R attended college, they are valid skips on this variable since we do not have dates for completion of college terms.
        However, if GRSURV=12 then they could reasonably have a value in this variable since in all cases June is
        PRIOR to the survey date, thus those who attended college whose grsurv=12 get a -3 on this variable.*/
if encat in (4,5,6,7,9,10,11,12) and grsurv>12 then grjune=-4;

/*Cases where R reported a higher grade in e3112 (HGC) than was reported in R3 AND who indicate their e2857
        (HGA) is at least two greater than the grsurv3 value, but who indicated that they were not enrolled should be
        coded as a -3 on both grsurv and grjune since the R is providing conflicting information*/
if enroll=0 and (grsurv=grsprev) and 12=>e3112>grsurv and e2857>0 and e2857-2>=grsurv then do; grsurv=-3; grjune=-
        3; encat=-3; assign='Conflicting information'; end;
```

```
/*************** HIGHEST DEGREE RECEIVED AS OF THE SURVEY DATE (DEGSURV)**************/
degsurv=0;
/* GED    */                  if ged=1 then degsurv=1; else if ged=-3 then degsurv=-3;
/* High school diploma */  if dip=1 then degsurv=2; else if dip=-3 then degsurv=-3;
if (e234501=1 or e234502=1 or e234503=1) then degsurv=3; /* Junior college or two-year associate degree */
   else if (e234501 in (-1, -2, -3) or e234502 in (-1, -2, -3) or e234503 in (-1, -2, -3)) then degsurv=-3;
if (e234501=3 or e234502=3 or e234503=3) then degsurv=4; /* Bachelor's degree */
   else if (e234501 in (-1, -2, -3) or e234502 in (-1, -2, -3) or e234503 in (-1, -2, -3)) then degsurv=-3;
if (e234501=4 or e234502=4 or e234503=4) then degsurv=5; /* Master's degree */
   else if (e234501 in (-1, -2, -3) or e234502 in (-1, -2, -3) or e234503 in (-1, -2, -3)) then degsurv=-3;
if (e234501=5 or e234502=5 or e234503=5) then degsurv=6; /* Doctoral degree */
   else if (e234501 in (-1, -2, -3) or e234502 in (-1, -2, -3) or e234503 in (-1, -2, -3)) then degsurv=-3;
if (e234501=6 or e234502=6 or e234503=6) then degsurv=7; /* Professional degree */
   else if (e234501 in (-1, -2, -3) or e234502 in (-1, -2, -3) or e234503 in (-1, -2, -3)) then degsurv=-3;


/*************** HIGHEST DEGREE RECEIVED AS OF JUNE 30, 2000 (DEGJUNE)**************/
dtma=0; dtdoc=0; dtprof=0; degjune=0;
/*  GED              */          if dtged gt 0 and dtged le 246 then degjune=1; else if dtged in (-3, -2, -1) then degjune=-3;
/*  High school diploma */ if dths gt 0 and dths le 246 then degjune=2; else if dths in (-3, -2, -1) then degjune=-3;
/* Junior college or 2-year associate degree */ if dtaa gt 0 and dtaa le 246 then degjune=3; else if dtaa in (-3, -2, -1) then
          degjune=-3;
/* Bachelor's degree */        if dtba gt 0 and dtba le 246 then degjune=4; else if dtba in (-3, -2, -1) then degjune=-3;
/* Master's degree */          if dtma gt 0 and dtma le 246 then degjune=5; else if dtma in (-3, -2, -1) then degjune=-3;
/* Doctoral degree */          if dtdoc gt 0 and dtdoc le 246 then degjune=6; else if dtdoc in (-3, -2, -1) then degjune=-3;
/* Professional degree */      if dtprof gt 0 and dtprof le 246 then degjune=7; else if dtprof in (-3, -2, -1) then degjune=-3;
if e3112 eq -5 then do; degsurv=-5; degjune=-5; end;
if 0=<degsurv=degsurv3 and degjune^=degsurv3 then degjune=degsurv3;

/***********HAND EDITS***********/
*R does not have HS diploma but is enrolled in HS diploma/AA program at comm. college;
    if id=1904482 then grsurv=13;
*case where interview date appears to have improperly been filled in for grade completion date;
    if id=1025602 then grsurv=11;
*Hand edit cases where no college is reported, but loop information identifies a GRSURV that is >12.  GRSURV is set to
          12 since there is no report of a college.  These cases are identified by the code: where 95>grsurv>12 and encat in
          (1,2,3,8);
if id in (1029001, 1044771, 1087541, 1135131, 1192241, 1268141, 1332691, 1471471, 1480891, 1632991, 1668581,
          1782401, 1933841) then do; grsurv=12; end;
/*still enrolled in 12th grade at int date*/
   if id=1404481 then grsurv=11;
/*loop grade is incorrectly recorded as 1*/
   if id=1048301 then grjune=-3;
/*inconsistent report of grades--indicates GED completion in comments, set to 12*/
   if id=1693361 then grsurv=12;

if grjune=. then grjune=-3;
if ged=1 and dtged<=0 then dateflag=1;
if dip=1 and dths<=0 then dateflag=1;
if dateflag=1 and degjune>=0 then degjune=-3;
if grsurv=-1 then grsurv=-3;

endsas;
```

CURRENT OR MOST RECENT SCHOOL PRIVATE OR PAROCHIAL

Variables Created: CV_SCHOOL_TYPE

Variables Used

Name in Program	Question Name on CD	Name in Program	Question Name on CD
typer1-typer3	CV_SCHOOL_TYPE (rounds 1-3)	sstop3m1, sstop3y1	NEWSCHOOL_STOP3.01~M, ~Y
speriod1-speriod7	NEWSCHOOL_PERIODS.01-.07	sstop3m2, sstop3y2	NEWSCHOOL_STOP3.02~M, ~Y
sstop1m1-sstop1m7	NEWSCHOOL_STOP1.01~M-.07~M	scode1-scode7	NEWSCHOOL_SCHCODE.01-.07
sstop1y1-sstop1y7	NEWSCHOOL_STOP1.01~Y-.07~Y	type1-type7	NEWSCHOOL_TYPE.01-.07
sstop2m1-sstop2m4	NEWSCHOOL_STOP2.01~M-.04~M	pubid	PUBID
sstop2y1-sstop2y4	NEWSCHOOL_STOP2.01~Y-.04~Y		

Codes for Created Variable

1 = Public school 3 = Parochial
2 = Private, not parochial 4 = Other

This program creates a variable indicating whether the respondent's current or most recent school is public, private, parochial, or other. The program first determines which school is the most recent school (non-college) by examining the end dates for different enrollment periods and then picks up the school type from rounds 1-4 for the most recent school. Users should note that school type is defined only for respondents attending grades 1-12.

```
/* Create the array variables.*/
* # of enrollment periods;                                array speriod speriod1-speriod5;
* ending date for each period;                            array sstop1m sstop1m1-sstop1m5;
                                                          array sstop1y sstop1y1-sstop1y5;
                                                          array sstop2m sstop2m1-sstop2m5;
                                                          array sstop2y sstop2y1-sstop2y5;
* school code, i.e. elementary school, high school or college;  array scode scode1-scode5;
*school type for round 4;                                 array stype type1-type5;

/*this gets rid of some duplicate school problems which remained in the roster after cleaning*/
do i=1 to 5;
  if scode[i]=4 or scode[i]=5 then do;
    if stype[i] ne -4 then do; stype[i]=-4; end;
  end;
  if scode[i] eq -4 and stype[i] ne -4 then do; stype[i]=-4; end;
end;

* ending time for each school;                             array srecm srecm1-srecm5;
                                                          array srecy srecy1-srecy5;
* school length counting both month&year: the month length from jan. 1990 to the ending month for each school;
                                                          array mlength mlength1-mlength5;
* school length counting only year: the year length from 1990 to the ending year for each school;
* this variable is to take care of the situation where ending year is invalid;
                                                          array ylength ylength1-ylength5;

/************************* step 1).find the most recent school. ***************************/
do i=1 to 5;
  srecm[i]=-4; srecy[i]=-4;
  if speriod[i]=1 and scode[i] in (1,2,3) then do; srecm[i]=sstop1m[i]; srecy[i]=sstop1y[i]; end;
  if speriod[i]=2 and scode[i] in (1,2,3) then do; srecm[i]=sstop2m[i]; srecy[i]=sstop2y[i]; end;
  if srecm[i] gt 0 and srecy[i] gt 0 then mlength[i]=12*(srecy[i]-1990)+srecm[i];
  else mlength[i]=-4;
  if srecy[i] gt 0 then ylength[i]=srecy[i]-1990;
  else ylength[i]=-4;
```

```
end;

maxmleng=max(mlength1,mlength2, mlength3, mlength4, mlength5);
if type1=-5 then maxmleng=-5;
maxyleng=max(ylength1,ylength2, ylength3, ylength4, ylength5);
if type1=-5 then maxyleng=-5;

/************* step 2).decide the school type from round 4 or the previous rounds.************/
sltype1=-4;
* use only the year information first, then use the month information to make it more precise;
do i=1 to 5; if maxyleng>0 and ylength[i]=maxyleng then sltype1=stype[i]; end;
do i=1 to 5; if maxmleng>0 and mlength[i]=maxmleng then sltype1=stype[i]; end;

* flag the cases where multiple schools' ending year are the same as "maxyleng" and ending months are invalid answer;
flag=0;flag1=0;
do i=1 to 5;
  if ylength[i]=maxyleng and srecm[i] in (-1,-2,-3) then flag=flag+1;
  if ylength[i]=maxyleng then flag1=flag1+1;
end;

/* go back to fix the case such as 97.12 and 99.-2. because if "maxmleng" cover "maxyleng", the codes above will pick up
        the school end on 97.12 as most recent school.*/
if flag1=1 then do;
  do i=1 to 5; if ylength[i]=maxyleng then sltype1=stype[i]; end;
end;
* for the invalid answers;
if scode1 in (-1,-2,-3) or scode2 in (-1,-2,-3) or scode3 in (-1,-2,-3) or scode4 in (-1,-2,-3) or scode5 in (-1,-2,-3) then
        sltype1=-3;
if (scode1 in (1,2,3) & srecy1 in (-1,-2,-3)) | (scode2 in (1,2,3) & srecy2 in (-1,-2,-3)) | (scode3 in (1,2,3) & srecy3 in (-
        1,-2,-3)) | (scode4 in (1,2,3) & srecy4 in (-1,-2,-3)) | (scode5 in (1,2,3) & srecy5 in (-1,-2,-3)) then sltype1=-3;
if flag ge 2 then sltype1=-3;
* fix the case such as 99.12, 99.-2 99.-3 or 2000.12 2000.-2 for the above line;
if flag1 gt 1 then do;
  do i=1 to 5; if ylength[i]=maxyleng and srecm[i]=12 then sltype1=stype[i]; end;
end;
* flag the cases which has 98.02 and 98.-3;
flag2=0; if flag1 ge 2 and flag ge 1 then flag2=1;
* if the respondent only goes to one school in r4,input that school type;
if scode1 in (1,2,3) and scode2 in (-4, 4, 5) and scode3 in (-4, 4, 5) and scode4 in (-4, 4, 5) and scode5 in (-4, 4, 5) then
        sltype1=type1;
*these are r's who are only in college in round 4 - used to correct missings;
col_ck=0; if scode1=4 or scode1=5 then col_ck=1;
*these are r's who don't report any r4 schools - they have dropped out (also used below);
drop=0; if type1 eq -4 and type2 eq -4 and type3 eq -4 and type4 eq -4 and type5 eq -4 then drop=1;
* use the previous rounds' info. to determine type if missing;
mstyp=0;
if type1=-4 and sltype1 ne -5 and col_ck=0 and drop=0 then do;
   if typer1 not in (-4,-5) then sltype1=typer1;        /* starting from round 1 */
   if typer2 not in (-4,-5) then sltype1=typer2;        /* round2's info. cover round1's info. */
   if typer3 not in (-4,-5) then sltype1=typer3;        /* round3's info. cover round2's info. */
   mstyp=1;
end;

/* recode the type variable.*/ sltype=-4;
if sltype1=1 then sltype=1;          if sltype1=5 then sltype=2;
if sltype1 in (3,4) then sltype=3;   if sltype1=2 or sltype1>=6 then sltype=4;
if -4<sltype1<0 then sltype=-3;      if type1=-5 then sltype=-5;                    endsas;
```

DATE RECEIVED DIPLOMA OR DEGREE

Variables Created: CV_GED CV_HS_DIPLOMA
CV_AA_DEGREE CV_BA_DEGREE

Variables Used

Name in Program	Question Name on CD	Name in Program	Question Name on CD
pubid	PUBID	s13500m s13500y	YSCH-13500~M, ~Y
gedr1, gedr2	CV_GED (rounds 1 and 2)	s234501-s234503	YSCH-23450.01-.03
hsr1, hsr2	CV_HS_DIPLOMA (rounds 1 and 2)	s234601m, s234601y	YSCH-23460.01~M, ~Y
s234501-s234504	YSCH-23450.01-.04 (round 2)	s234602m, s234602y	YSCH-23460.02~M, ~Y
s11700	YSCH-11700	s234603m, s234603y	YSCH-23460.03~M, ~Y
s11900m, s11900y	YSCH-11900~M, ~Y	t7000_1-t7000_4	YTRN-7000.01-.04
s13300	YSCH-13300		

This program creates several continuous month variables, identifying the month that the respondent received a GED, high school diploma, associate's degree, and bachelor's degree. In future rounds, similar variables will calculate the month a master's, doctoral, or professional degree was received; no respondents had received these degrees in rounds 1–4. For more information on the continuous month scheme, see appendix 7 in this document.

```
/* First, create the date variables using round 4 information only.*/

hsr4=-4;        gedr4=-4;       aar4=-4;        bar4=-4;
mar4=-4;        docr4=-4;       profr4=-4;      flag=0;

if s11700=1 and s11900m gt 0 and s11900y gt 0 then hsr4=(s11900y-1980)*12+s11900m;
if -4<s11700<0 then hsr4=s11700;
if -4<s11900m<0 then hsr4=s11900m;
if -4<s11900y<0 then hsr4=s11900y;

if s13300=1 and s13500m>0 and s13500y>0 then gedr4=(s13500y-1980)*12+s13500m;
if -4<s13300<0 then gedr4=s13300;
if -4<s13500m<0 then gedr4=s13500m;
if -4<s13500y<0 then gedr4=s13500y;

array s23450 s234501-s234503;
array s23460m s234601m s234602m s234603m;
array s23460y s234601y s234602y s234603y;
array t7000 t7000_1 t7000_2 t7000_3 t7000_4;

do i=1 to 3;
if s23450[i]=1 and s23460m[i]>0 and s23460y[i]>0 and aar4=-4 then aar4=(s23460y[i]-1980)*12+s23460m[i];
if s23450[i]=1 and -4<s23460m[i]<0 and aar4=-4 then aar4=s23460m[i];
if s23450[i]=1 and -4<s23460y[i]<0 and aar4=-4 then aar4=s23460y[i];
if s23450[i]=3 and s23460m[i]>0 and s23460y[i]>0 and bar4=-4 then bar4=(s23460y[i]-1980)*12+s23460m[i];
if s23450[i]=3 and -4<s23460m[i]<0 and bar4=-4 then bar4=s23460m[i];
if s23450[i]=3 and -4<s23460y[i]<0 and bar4=-4 then bar4=s23460y[i];
if s23450[i]=4 and s23460m[i]>0 and s23460y[i]>0 and mar4=-4 then mar4=(s23460y[i]-1980)*12+s23460m[i];
if s23450[i]=4 and -4<s23460m[i]<0 and mar4=-4 then mar4=s23460m[i];
if s23450[i]=4 and -4<s23460y[i]<0 and mar4=-4 then mar4=s23460y[i];
if s23450[i]=5 and s23460m[i]>0 and s23460y[i]>0 and docr4=-4 then docr4=(s23460y[i]-1980)*12+s23460m[i];
if s23450[i]=5 and -4<s23460m[i]<0 and docr4=-4 then docr4=s23460m[i];
if s23450[i]=5 and -4<s23460y[i]<0 and docr4=-4 then docr4=s23460y[i];
if s23450[i]=6 and s23460m[i]>0 and s23460y[i]>0 and profr4=-4 then profr4=(s23460y[i]-1980)*12+s23460m[i];
if s23450[i]=6 and -4<s23460m[i]<0 and profr4=-4 then profr4=s23460m[i];
if s23450[i]=6 and -4<s23460y[i]<0 and profr4=-4 then profr4=s23460y[i];
if s23450[i]=9 and s23460m[i]>0 and s23460y[i]>0 and gedr4=-4 then gedr4=(s23460y[i]-1980)*12+s23460m[i];
```

```
if s23450[i]=9 and -4<s23460m[i]<0 and gedr4=-4 then gedr4=s23460m[i];
if s23450[i]=9 and -4<s23460y[i]<0 and gedr4=-4 then gedr4=s23460y[i];
if s23450[i]=8 or -4<s23450[i]<0 then flag=1;

if -4<s23450[i]<0 and aar4=-4 then aar4=s23450[i];
if -4<s23450[i]<0 and mar4=-4 then mar4=s23450[i];
if -4<s23450[i]<0 and profr4=-4 then profr4=s23450[i];
if -4<s23450[i]<0 and bar4=-4 then bar4=s23450[i];
if -4<s23450[i]<0 and docr4=-4 then docr4=s23450[i];
if -4<s23450[i]<0 and gedr4=-4 then gedr4=s23450[i];

end;

/*if respondent completes GED in a training program, set date to -3 because we don't know completion date.*/
ged_tr=0;
do i=1 to 4;
  if gedr4=-4 and t7000[i]=4 then do; gedr4=-3; ged_tr=1; end;
end;
if s11700=-5 then do; gedr4=-5; hsr4=-5; aar4=-5; bar4=-5; mar4=-5; docr4=-5; profr4=-5; end;

/*change -2 to -3 for those who don't know completion date.*/
array created aar4 bar4 mar4 docr4 profr4 gedr4;
do i=1 to 6; if created[i] = -2 then created[i]=-3; end;

/* second, we will check if there is some conflict between round 3 and round 4.*/
flaghs=0;
flagged=0;
if hsr4 not in (-4,-5) and hsr3 not in  (-4,-5) and hsr4 ne hsr3 then flaghs=1;
if gedr4 not in (-4,-5) and gedr3 not in (-4,-5) and gedr4 ne gedr3 then flagged=1;

/* at last, we will merge the result from round 2 and round 1.*/
dtged=-4; dths=-4; dtaa=-4; dtba=-4; dtma=-4; dtdoc=-4; dtprof=-4;

if flagged ne 1 then do;
  if gedr3 not in (-4,-5) then dtged=gedr3;
  if gedr4 ne -4 then dtged=gedr4;
end;
if flagged eq 1 then do;
  if gedr3 eq -3 and gedr4 gt 0 then do; dtged=gedr4; flagged=0; end;
end;

/*hand edit due to interviewer comment*/
if id=1446573 then do; dtged=240; flagged=0; end;
/*hand edit due to r give different ged/hs degree dates.  used first date since closer to event and checks out in logs*/
if id=1518151 then do; dths=210; flaghs=0; end;
if id=1021941 then do; dtged=205; flagged=0; end;
if id=1842201 then do; dtged=212; flagged=0; end;

if flaghs ne 1 then do;
  if hsr3 not in (-4,-5) then dths=hsr3;
  if hsr4 ne -4 then dths=hsr4;
  if dths=-2 then dths=-3;
end;

/*account for previously earned assoc. & ba degrees (this needs to be added to program)*/
if id=1079191 then aar4=240; if id=1644591 then aar4=-3; if id=1874592 then bar4=-3;

dtaa=aar4; dtba=bar4; dtma=mar4; dtdoc=docr4; dtprof=profr4;
```

Number of Grades Repeated or Skipped

Variables Created: CV_GRADES_REPEAT_EVER CV_GRADES_REPEAT_YR
CV_GRADE_SKIPPED_EVER CV_GRADE_SKIPPED_YR

Variables Used

Name in Program	Question Name on CD
Round 1	
r1intday, r1intmo, r1intyr	YINF-900_D, _M, _Y
s3500r1	YSCH-3500
cvrpevr1	CV_GRADES_REPEAT_EVER
cvskevr1	CV_GRADE_SKIPPED_EVER
Round 2	
s2857r2	YSCH-2857
r2intday, r2intmo, r2intyr	SYMBOL!CURDATE~D, ~M, ~Y
cvrpevr2	CV_GRADES_REPEAT_EVER
cvskevr2	CV_GRADE_SKIPPED_EVER
Round 3	
s2857r3	YSCH-2857
r3intday, r3intmo, r3intyr	SYMBOL!CURDATE~D, ~M, ~Y
cvrpevr3	CV_GRADES_REPEAT_EVER
cvskevr3	CV_GRADE_SKIPPED_EVER
Round 4	
r4intday, r4intmo, r4intyr	SYMBOL!CURDATE~D, ~M, ~Y
s3104	YSCH-3104
s2857	YSCH-2857
s4846m, s4846y	YSCH-4846~M, ~Y
s162001m, s162001y	YSCH-1620.01~M, ~Y
s162701	YSCH-1627.01
s4951	YSCH-4951
s841611, s841612	YSCH-8416.01.01, .02
s842621, s841622	YSCH-8416.02.01, .02
s841631, s841632	YSCH-8416.03.01, .02
s841641, s841651	YSCH-8416.04.01, .05.01
s877411m, s877411y	YSCH-8774.01.01~M, ~Y
s877412m, s877412y	YSCH-8774.01.02~M, ~Y
s877421m, s877421y	YSCH-8774.02.01~M, ~Y
s877422m, s877422y	YSCH-8774.02.02~M, ~Y
s877431m, s877431y	YSCH-8774.03.01~M, ~Y
s877432m, s877432y	YSCH-8774.03.02~M, ~Y
s877441m, s877441y	YSCH-8774.04.01~M, ~Y
s897711, s897712	YSCH-8977.01.01, .02
s897721, s897722	YSCH-8977.02.01, .02
s897731, s897732	YSCH-8977.03.01, .02
s897741	YSCH-8977.04.01
s913111 s913112	YSCH-9131.01.01, .02
s913121 s913122	YSCH-9131.02.01, .02
s913131 s913132	YSCH-9131.03.01, .02
s913141 s913151	YSCH-9131.04.01, .05.01
s933511m, s933511y	YSCH-9335.01.01~M, ~Y
s933512m, s933512y	YSCH-9335.01.02~M, ~Y
s933521m, s933521y	YSCH-9335.02.01~M, ~Y
s933522m, s933522y	YSCH-9335.02.02~M, ~Y
s933531m, s933531y	YSCH-9335.03.01~M, ~Y
s933541m, s933541y	YSCH-9335.04.01~M, ~Y
s933551m, s933551y	YSCH-9335.05.01~M, ~Y
s9589111-s9589114	YSCH-9589.01.01.01-.04
s9589121	YSCH-9589.01.02.01
s9589211-s9589213	YSCH-9589.02.01.01-.03
s9589221	YSCH-9589.02.02.01
s9589311, s9589312	YSCH-9589.03.01.01, .02
s9589411, s9589511	YSCH-9589.04.01.01, .05.01.01
s691111m, s691111y	YSCH-9691.01.01.01~M, ~Y
s691112m, s691112y	YSCH-9691.01.01.02~M, ~Y
s691113m, s691113y	YSCH-9691.01.01.03~M, ~Y
s691114m, s691114y	YSCH-9691.01.01.04~M, ~Y
s691121m, s691121y	YSCH-9691.01.02.01~M, ~Y
s691211m, s691211y	YSCH-9691.02.01.01~M, ~Y
s691212m, s691212y	YSCH-9691.02.01.02~M, ~Y
s691213m, s691213y	YSCH-9691.02.01.03~M, ~Y
s691221m, s691221y	YSCH-9691.02.02.01~M, ~Y
s691311m, s691311y	YSCH-9691.03.01.01~M, ~Y
s691312m, s691312y	YSCH-9691.03.01.02~M, ~Y
s691411m, s691411y	YSCH-9691.04.01.01~M, ~Y
s691511m, s691511y	YSCH-9691.05.01.01~M, ~Y
s9793111-s9793113	YSCH-9793.01.01.01-.03
s9793211, s9793212	YSCH-9793.02.01.01, .02
s9793311	YSCH-9793.03.01.01
s9946111-s9946114	YSCH-9946.01.01.01-.04
s9946121	YSCH-9946.01.02.01
s9946211-s9946213	YSCH-9946.02.01.01-.03
s9946221	YSCH-9946.02.02.01
s9946311, s9946312	YSCH-9946.03.01.01, .02
s9946411, s9946511	YSCH-9946.04.01.01, .05.01.01
s099111m, s099111y	YSCH-10099.01.01.01~M, ~Y
s099112m, s099112y	YSCH-10099.01.01.02~M, ~Y
s099113m, s099113y	YSCH-10099.01.01.03~M, ~Y
s099114m, s099114y	YSCH-10099.01.01.04~M, ~Y
s099211m, s099211y	YSCH-10099.02.01.01~M, ~Y
s099212m, s099212y	YSCH-10099.02.01.02~M, ~Y
s099311m, s099311y	YSCH-10099.03.01.01~M, ~Y
s099312m, s099312y	YSCH-10099.03.01.02~M, ~Y
strt11m, strt11y	NEWSCHOOL_START1.01~M, ~Y
strt12m, strt12y	NEWSCHOOL_START1.02~M, ~Y
strt13m, strt13y	NEWSCHOOL_START1.03~M, ~Y
strt14m, strt14y	NEWSCHOOL_START1.04~M, ~Y
strt15m, strt15y	NEWSCHOOL_START1.05~M, ~Y
id	PUBID
prehga	SYMBOL!REVPREV!HGA
enroll	Enrolled at int date r4 - created
stop11m, stop11y	NEWSCHOOL_STOP1.01~M, ~Y
stop12m, stop12y	NEWSCHOOL_STOP1.02~M, ~Y
stop13m, stop13y	NEWSCHOOL_STOP1.03~M, ~Y
stop14m, stop14y	NEWSCHOOL_STOP1.04~M, ~Y
stop15m, stop15y	NEWSCHOOL_STOP1.05~M, ~Y
stop21m, stop21y	NEWSCHOOL_STOP2.01~M, ~Y
stop22m, stop22y	NEWSCHOOL_STOP2.02~M, ~Y
stop23m, stop23y	NEWSCHOOL_STOP2.03~M, ~Y
scode1-scode5	NEWSCHOOL_SCHCODE.01-.05

This program creates variables counting the number of grades the respondent ever repeated or skipped. The first variable in each pair counts the total number of grades ever repeated/skipped through the interview date; the second variable calculates the number through June 30, 2000.

```
/******Create new array of start dates for first school depending on whether r. corrected roster data******/
array strt1xm strt11m strt12m strt13m strt14m strt15m strt16m strt17m;
array strt1xy strt11y strt12y strt13y strt14y strt15y strt16y strt17y;
array s8771m s877411m s877421m s877431m s877441m s877451m s877461m s877471m;
array s8771y s877411y s877421y s877431y s877441y s877451y s877461y s877471y;
array startm start1m start2m start3m start4m start5m start6m start7m;
array starty start1y start2y start3y start4y start5y start6y start7y;

*Hand edit for respondents whose reported third school is actually the first school;
if id in (1035781,1193561) then do;
  s841631=-4; s9131=-4; s897711=s897731; s897731=-4; strt11m=strt13m; strt13m=-4; strt11y=strt13y; strt13y=-4;
end;

numstart=0;
do i = 1 to 7;
  if s8771m[i] gt 0 and s8771y[i] gt 0 then do; startm[i]=s8771m[i]; starty[i]=s8771y[i]; end;
  else if s8771m[i] le 0 or s8771y[i] le 0 then do; startm[i]=strt1xm[i]; starty[i]=strt1xy[i]; numstart=numstart+1; end;
end;

*******Create arrays of grades attended, whether they were completed, and start and completion dates;
***grade attended;
array gr_att s841611 s841612 s841613 s841621 s841622 s841623 s841631 s841632 s841641 s841642 s841651
        s841661 s841671 s9589111 s9589112 s9589113 s9589114 s9589121 s9589211 s9589212 s9589213 s9589221
        s9589311 s9589312 s9589411 s9589511;
***Completed y/n;
array gr_comp s913111 s913112 s913113 s913121 s913122 s913123 s913131 s913132 s913141 s913142 s913151
        s913161 s913171 s9946111 s9946112 s9946113 s9946114 s9946121 s9946211 s9946212 s9946213 s9946221
        s9946311 s9946312 s9946411 s9946511;
***When completed grade - month;
array cdate_m s933511m s933512m s933513m s933521m s933522m s933523m s933531m s933532m s933541m
        s933542m s933551m s933561m s933571m s099111m s099112m s099113m s099114m s099121m s099211m
        s099212m s099213m s099221m s099311m s099312m s099411m s099511m;
***When completed grade - year;
array cdate_y s933511y s933512y s933513y s933521y s933522y s933523y s933531y s933532y s933541y s933542y
        s933551y s933561y s933571y s099111y s099112y s099113y s099114y s099121y s099211y s099212y
        s099213y s099221y s099311y s099312y s099411y s099511y;
***When started - month;
array s_date_m start1m s877412m s877413m start2m s877422m s877423m start3m s877432m start4m s877442m
        start5m start6m start7m s691111m s691112m s691113m s691114m s691121m s691211m s691212m
        s691213m s691221m s691311m s691312m s691411m s691511m;
***When started - year;
array s_date_y start1y s877412y s877413y start2y s877422y s877423y start3y s877432y start4y s877442y start5y start6y
        start7y s691111y s691112y s691113y s691114y s691121y s691211y s691212y s691213y s691221y s691311y
        s691312y s691312y s691511y;
*****Create arrays of all grades attended and completed and their start and complete dates;
array attend attend1-attend9;                array comp comp1-comp9;
array datestm datestm1-datestm9;             array datesty datesty1-datesty9;
array datecpm datecpm1-datecpm9;             array datecpy datecpy1-datecpy9;

****Set all new array vars to -4;
do i=1 to 9;
   datestm[i]=-4; datesty[i]=-4; datecpm[i]=-4; datecpy[i]=-4; comp[i]=-4; attend[i]=-4;
end;
```

```
*****Record all grades completed and their completion dates;
j=1;
do i= 1 to 26;
   if gr_comp[i] = 1 and gr_att[i] not in (-4,-5) then do;
      comp(j)=gr_att[i]; datecpm(j)=cdate_m[i]; datecpy(j)=cdate_y[i]; j=j+1;
   end;
end;

*****Record all grades attended and their start dates;
k=1;
flag1=0;
if s913111 not in (-4,-5) then do;
   if s841611 not in (-4,-5) then do;
      attend[k]=s841611;
      if start1m>0 then datestm[k]=start1m;
      if start1y>0 then datesty[k]=start1y;
      if s877411m=-4 and s877411y=-4 and strt11m=-4 and strt11y=-4 then flag1=1;
      k=k+1;
   end;
   do i=2 to 25;
      if gr_comp[i] ne -4 and gr_comp[i] ne . then do;
        attend(k)=gr_att[i]; datestm(k)=s_date_m[i]; datesty(k)=s_date_y[i]; k=k+1;
      end;
   end;
end;

*******Create date in continuous months for each start and complete date in arrays;
array c_date c_date1 c_date2 c_date3 c_date4 c_date5 c_date6 c_date7 c_date8 c_date9;
array s_date s_date1 s_date2 s_date3 s_date4 s_date5 s_date6 s_date7 s_date8 s_date9;
do i=1 to 9;
      if datecpy[i] gt 0 and datecpm[i] gt 0 then c_date[i]=((datecpy[i]-1980)*12)+datecpm[i];
      else if datecpy[i] in (-1, -2, -3) or datecpm[i] in (-1, -2, -3) then c_date[i]=-3;
      else if datecpy[i] eq -4 or datecpm[i] eq -4 then c_date[i]=-4;
      if datesty[i] gt 0 and datestm[i] gt 0 then s_date[i]=((datesty[i]-1980)*12)+datestm[i];
      else if datesty[i] in (-1, -2, -3) or datestm[i] in (-1, -2, -3)  then s_date[i]=-3;
      else if datesty[i] eq -4 or datestm[i] eq -4 then s_date[i]=-4;
end;

*****Create new arrays of started and completed grades that have grades listed only once and in order- report latest
            complete date;
array newcomp newcomp1 newcomp2 newcomp3 newcomp4;
array newcdate newcdat1 newcdat2 newcdat3 newcdat4;

newcomp1=comp1;
newcdat1=c_date1;

***First make sure that the first element in the array is the lowest grade and the latest date for that grade;
do i=2 to 4;
   if comp[i] gt 0 then do;
      if comp[i] eq newcomp1 and c_date[i] gt newcdat1 then newcdat1=c_date[i];
      else if comp[i] lt newcomp1 then do; newcomp1=comp[i]; newcdat1=c_date[i]; end;
   end;
end;

j=2;
do i=1 to 4;
```

```
  if newcomp(j-1) ne . then do;
    if comp[i] eq newcomp(j-1) and c_date[i] gt newcdate(j-1) then newcdate(j-1)=c_date[i];
    else if comp[i] eq newcomp(j-1) + 1 then do; newcomp(j)=comp[i]; newcdate(j)=c_date[i]; end;
    else if newcomp(j)=. and comp[i] gt newcomp(j-1) then do; newcomp(j)=comp[i]; newcdate(j)=c_date[i]; end;
  end;
end;

j=3;
do i=1 to 4;
  if newcomp(j-1) ne . then do;
    if comp[i] eq newcomp(j-1) and c_date[i] gt newcdate(j-1) then newcdate(j-1)=c_date[i];
    else if comp[i] eq newcomp(j-1) + 1 then do; newcomp(j)=comp[i]; newcdate(j)=c_date[i]; end;
    else if newcomp(j)=. and comp[i] gt newcomp(j-1) then do; newcomp(j)=comp[i]; newcdate(j)=c_date[i]; end;
  end;
end;

j=4;
do i=1 to 4;
  if newcomp(j-1) ne . then do;
    if comp[i] eq newcomp(j-1) and c_date[i] gt newcdate(j-1) then newcdate(j-1)=c_date[i];
    else if comp[i] eq newcomp(j-1) + 1 then do; newcomp(j)=comp[i]; newcdate(j)=c_date[i]; end;
    else if newcomp(j)=. and comp[i] gt newcomp(j-1) then do; newcomp(j)=comp[i]; newcdate(j)=c_date[i]; end;
  end;
end;

***Create a new array with all grades attended in order and an array with the earliest date each grade was started;
array tmpatt tmpatt1 tmpatt2 tmpatt3 tmpatt4 tmpatt5 tmpatt6 tmpatt7 tmpatt8 tmpatt9;
array tmpadate tmpadat1 tmpadat2 tmpadat3 tmpadat4 tmpadat5 tmpadat6 tmpadat7 tmpadat8 tmpadat9;
array newatt newatt1 newatt2 newatt3 newatt4 newatt5 newatt6 newatt7 newatt8 newatt9;
array newadate newadat1 newadat2 newadat3 newadat4 newadat5 newadat6 newadat7 newadat8 newadat9;

do i=1 to 9; tmpatt[i]=attend[i]; tmpadate[i]=s_date[i]; end;
j=1;
i=1;
k=1;
do i=1 to 9;
    do j=i+1 to 9;
      if tmpatt[j] ne -4 then do;
         if tmpatt[j] lt tmpatt[i] and tmpatt[i] gt 0 and tmpatt[j] gt 0 then do;
              att_temp = tmpatt[i]; dat_temp = tmpadate[i]; tmpatt[i]=tmpatt[j]; tmpadate[i] = tmpadate[j]; tmpatt[j] =
                      att_temp; tmpadate[j] = dat_temp;
          end;
       end;
    end;
    if i eq 1 then do; newatt[k] = tmpatt[i]; newadate[k] = tmpadate[i]; end;
    if tmpatt[i] eq newatt[k] then do;
         if tmpadate[i] lt newadate[k] then do; newatt[k] = tmpatt[i]; newadate[k] = tmpadate[i]; end;
    end;
    else if tmpatt[i] gt newatt[k] then do; k=k+1; newatt[k] = tmpatt[i]; newadate[k] = tmpadate[i]; end;
    j=i+1;
end;

***Find last enrollment date;
array stopm stop11m stop21m stop12m stop22m stop13m stop23m stop14m stop15m;
array stopy stop11y stop21y stop12y stop22y stop13y stop23y stop14y stop15y;
array stopd stopd1-stopd8;
array scode scode1 scode1 scode2 scode2 scode3 scode3 scode4 scode5;
```

```
do i=1 to 8;
   if stopy[i] gt 0 and stopm[i] gt 0 then stopd[i]=((stopy[i]-1980)*12)+stopm[i];
   else if stopy[i]=-5 then stopd[i]=-5;
   else if stopy[i] in (-1,-2,-3) or stopm[i] in (-1,-2,-3) then stopd[i]=-3;
end;

if enroll=0 then do;
    stopdate=stopd1;
    do i=2 to 8; if scode[i] in (1,2,3) and stopd[i]>stopdate then stopdate=stopd[i]; end;
end;
else if enroll=1 then stopdate=-4;

****Create interview dates in continuous months;
if r1intyr gt 0 then r1intdte=((r1intyr-1980)*12)+r1intmo;
if r2intyr gt 0 then r2intdte=((r2intyr-1980)*12)+r2intmo;
else if r2intyr=-5 then r2intdte=-5;
if r3intyr gt 0 then r3intdte=((r3intyr-1980)*12)+r3intmo;
else if r3intyr=-5 then r3intdte=-5;
if r4intyr gt 0 then intdate=((r4intyr-1980)*12)+r4intmo;

prevint=r3intdte;
if prevint=-5 then prevint=r2intdte;
if prevint=-5 then prevint=r1intdte;

* Set end dates for attended grades: find the end date for each grade attended - if r. doesn't give a completion date, choose
            the month before they started the next grade;
array newedate newedat1 newedat2 newedat3 newedat4 newedat5 newedat6 newedat7 newedat8 newedat9;
array iscomp iscomp1 iscomp2 iscomp3 iscomp4 iscomp5 iscomp6 iscomp7 iscomp8 iscomp9;

do i=1 to 9; iscomp(i)=0; end;
endimp=0;
do i=1 to 9;
    do j=1 to 4;
      if newatt(i)=newcomp(j) and newatt(i) gt 0 then do; newedate(i)=newcdate(j); iscomp(i)=1; end;
    end;
    if newatt(i) ne . and newedate(i)=. and newadate(i) ne prevint and newadate(i) ne newadate(i+1) and newadate(i+1) ne
            . then do; newedate(i)=newadate(i+1)-1; endimp=1; end;
    else if newatt(i) ne . and newedate(i)=. and newatt(i+1)=. and enroll=1 then do; newedate(i)=intdate; endimp=1; end;
    else if newatt(i) ne . and newedate(i)=. and newatt(i+1)=. and enroll=0 then do; newedate(i)=stopdate; endimp=1; end;
    else if newedate(i) eq . and newadate(i) eq prevint then do;
        endimp=1;
        if newadate(i) le 210 then newedate(i)=210;
        else if 211 le newadate(i) le 222 then newedate(i)=222;
        else if 223 le newadate(i) le 234 then newedate(i)=234;
        else if 235 le newadate(i) le 246 then newedate(i)=246;
        else if 247 le newadate(i) le 258 then newedate(i)=258;
    end;
end;

********delete the grades whose information is repeatedly reported in round 4;
array r3nadate r3nadat1 r3nadat2 r3nadat3 r3nadat4 r3nadat5 r3nadat6 r3nadat7 r3nadat8 r3nadat9;
array r3natt r3natt1 r3natt2  r3natt3  r3natt4  r3natt5  r3natt6  r3natt7  r3natt8  r3natt9;
array r3nedate r3nedat1 r3nedat2 r3nedat3 r3nedat4 r3nedat5 r3nedat6 r3nedat7 r3nedat8 r3nedat9;
array r3ncomp r3ncomp1-r3ncomp9;
array r2nadate r2nadat1 r2nadat2 r2nadat3 r2nadat4 r2nadat5 r2nadat6 r2nadat7 r2nadat8 r2nadat9;
array r2natt r2natt1 r2natt2  r2natt3  r2natt4  r2natt5  r2natt6  r2natt7  r2natt8  r2natt9;
```

```
array r2nedate r2nedat1 r2nedat2 r2nedat3 r2nedat4 r2nedat5 r2nedat6 r2nedat7 r2nedat8 r2nedat9;
array r2ncomp r2ncomp1-r2ncomp9;
array preadate preadat1 preadat2 preadat3 preadat4 preadat5 preadat6 preadat7 preadat8 preadat9;
array preatt preatt1 preatt2  preatt3  preatt4  preatt5  preatt6  preatt7  preatt8  preatt9;
array preedate preedat1 preedat2 preedat3 preedat4 preedat5 preedat6 preedat7 preedat8 preedat9;
array precomp precomp1-precomp9;
array tmp2att tmp2att1-tmp2att9;

do i=1 to 9; tmp2att[i]=newatt[i]; end;

backcut=0;
ncut=0;
backrep=0;

do i=1 to 9;
  do j=1 to 9;
    if tmp2att[i]>0 and ((tmp2att[i]=r3natt[j]  and r3ncomp[j]=1) or  (tmp2att[i]=r2natt[j] and r2ncomp[j]=1)) then do;
      j=9;backcut=1;ncut=ncut+1;
      do k=i+1-ncut to 9-ncut;
        if newatt[k+1] ne . then do; newatt[k]=newatt[k+1]; newadate[k]=newadate[k+1]; newedate[k]=newedate[k+1];
              iscomp[k]=iscomp[k+1]; end;
        else do; newatt[k]=.; newadate[k]=.; newedate[k]=.; iscomp[k]=0; k=9-ncut; end;
      end;
    end;
  end;
end;


if newatt1>0 then do;
do i=1 to 8;
   if (r3natt[i] ge newatt1 and r3natt[i+1] ne .) or (r3natt[i] ge newatt1 and r3ncomp[i]=1) or (r2natt[i] ge newatt1 and
              r2natt[i+1] ne .) or (r2natt[i] ge newatt1 and r2ncomp[i]=1) then backrep=1;
   end;
end;

*****Substitute start dates in the previous round for first grade in loops;
pre2857=s2857r3;
do i=1 to 9; preatt[i]=r3natt[i]; preadate[i]=r3nadate[i]; preedate[i]=r3nedate[i]; precomp[i]=r3ncomp[i]; end;
   if r3intyr=-5 then do;
      pre2857=s2857r2;
      do i=1 to 9; preatt[i]=r2natt[i]; preadate[i]=r2nadate[i]; preedate[i]=r2nedate[i]; precomp[i]=r2ncomp[i]; end;
   end;
   if r3intyr=-5 and r2intyr=-5 then do;
      pre2857=-5;
      do i=1 to 9; preatt[i]=-5; preadate[i]=-5; preedate[i]=-5; precomp[i]=-5; end;
   end;
badpre=0;
replace=0;
jumprep1=0;
jumprep2=0;
jumpge21=0;
jumpge22=0;

*flag the case where the reported grades are jumping;
if preatt1 ne -5 then do;
   do i=1 to 8;
```

```
      if preatt[i] ne . and preatt[i+1] eq . then do; lphigh=preatt[i]; lphedat=preedate[i]; lphighcp=precomp[i]; lpnum=i;
            i=9; end;
    end;
end;
else lphigh=-5;

if (lphigh ne -5 and ((newatt1=lphigh+1 and lphighcp ne 1) or newatt1>lphigh+1)) then jumprep1=1;
if (lphigh=-5 and ((hgcr1>0 and newatt1 ge hgcr1+2) or (s3500r1>0 and newatt1 ge s3500r1+2)))  then jumprep2=1;
if (lphigh ne -5 and newatt1 ge lphigh+2) then jumpge21=1;
if (lphigh=-5 and hgcr1>0 and newatt1 ge hgcr1+3) then jumpge22=1;

if newatt1 gt 0 and (r3intyr ne -5 or r2intyr ne -5) then do;
   do i=1 to 8;
      if (preadate(i) gt preedate(i) and preedate(i) gt 0) or (preedate(i) gt preadate(i+1) and preadate(i+1) gt 0) or
            (preadate(i+1) gt preedate(i+1) and preedate(i+1) gt 0) then do;
         newadat1=-3;
         badpre=1;
      end;
      else if preatt(i)=newatt1 and preatt(i+1)=. and precomp[i]=0 and s3104 ne -5 and badpre ne 1 then do;
         newadat1=preadate(i);
         replace=1;
      end;
   end;
end;

else if -4 lt newatt1 le 0 then newadat1=-3;

***Measure the length that a respondent was in each completed grade and count the number of grades skipped and
            repeated;
array lengtha length1 length2 length3 length4 length5 length6 length7 length8 length9;
repeat=0; repeatj=0; skipleng=0; skipl_j=0; skip=-16; skipj=-16; oldstrt=0; nocomp=0;

do i=1 to 8;
   if newedate[i] gt 0 and newadate[i] gt 0 then lengtha[i]=newedate[i]-newadate[i];
   if newadate[i] le newedate[i] le newadate[i+1] and lengtha[i] gt 13 and newatt[i+1]-newatt[i]=1 and newatt[i] ne 95 and
            repeat ne -3 then do;
      repeat=repeat+1;
      if i=1 and newadate(i) lt prevint then oldstrt=1;
      if newedate(i) le 246 then repeatj=repeatj+1;
   end;
   else if newadate[i] le newedate[i] and newatt[i+1]=. and lengtha[i] gt 13 and newatt[i] ne 95 and repeat ne -3 and
            (iscomp[i]=1 or s2857=newatt[i]) then do;
       repeat=repeat+1;
       if i=1 and newadate(i) lt prevint then oldstrt=1;
       if newedate(i) le 246 then repeatj=repeatj+1;
   end;
   else if newadate[i] le newedate[i] and newatt[i+1]=. and lengtha[i] gt 13 and newatt[i] ne 95 and repeat ne -3 and
            (iscomp[i]=0 and s2857 ne newatt[i]) then do;
       if i=1 and newadate(i) lt prevint then oldstrt=1;
       repeat=-3;
       repeatj=-3;
   end;
end;

do i=1 to 7;
   if newadate(i) lt newedate(i) lt newadate(i+1) lt newedate(i+1) then do;
```

```
      if newadate(i) in (212, 213, 224, 225, 236, 237, 248, 249) and (newadate(i) ne prevint) and newatt(i+1)-newatt(i)=1
            and iscomp(i+1)=1 and newedate(i+1)-newadate(i) le 12  and newadate(i+2)-newedate(i+1) le 9 then do;
         if (i eq 1 or (i ne 1 and newadate(i)-newedate(i-1) le 9)) and skipleng ne -3 then do;
            skipleng=skipleng+1;
            if newedate(i+1) le 246 and skipl_j ne -3 then skipl_j=skipl_j+1;
         end;
      end;
   end;
end;

   *Take care of connection between previous round and current round;
   conect=0;
   conectok=0;
   if newatt1=lphigh and iscomp1=1 and lphighcp ne 1 then do; conect=1; if lpnum>2 then conectok=1; end;

***Check for respondents who completely skip (do not attend) a grade;
skipgr=0; skipgrj=0; skipgt2=0;
do i=1 to 8;
    if newatt(i+1)-newatt(i) eq 2 and newadate(i+1)-newedate(i) le 9 and skipgr ne -3 then do;
        skipgr=skipgr+1;
        if newedate(i+1) le 246 then skipgrj=skipgrj+1;
    end;
    else if newatt(i+1)-newatt(i) eq 3 and newadate(i+1)-newedate(i) le 9 then do;
        skipgr=skipgr+2;
        if newedate(i+1) le 246 then skipgrj=skipgrj+2;
    end;
    else if newatt(i+1)-newatt(i) gt 3 and newadate(i+1)-newedate(i) le 9 then do; skipgt2=1; end;
end;

***Set repeat and skip to -3 if grades out of order or if backrep=1 or jumprep=1 or if report 0 grade;
allmess=0;
do i=1 to 8;
   if (newadate(i) gt newedate(i) and newedate(i) gt 0) or (newedate(i) gt newadate(i+1) and newadate(i+1) gt 0) or
            (newadate(i+1) gt newedate(i+1) and newedate(i+1) gt 0) then do; allmess=1; end;
end;

zerograd=0;
do i=1 to 9; if newatt[i]=0 then zerogrd=1; end;

if allmess=1 or backrep=1 or zerograd=1 or jumprep1=1 or jumprep2=1 then do; repeat=-3; repeatj=-3; skip=-3; skipj=-3;
            end;

***Flag respondents who reported skipping a grade and those who say they did NOT skip grade;
array whyskip s897711 s897712 s897721 s897722 s897731 s897732 s897741 s9793111 s9793112 s9793113 s9793211
            s9793212 s9793311;
skipques=0;
do i=1 to 13; if whyskip[i]=1 then skipques=1; end;
notskip=0;
do i=1 to 13; if whyskip[i] in (-2,2,999) then notskip=1; end;

if skip ne -3 then do;
   if (skipques eq 0 and skipleng=0 and skipgr=0) or (notskip ge 1 and skipques=0) then skip=0;
   else if skipleng eq -3 and (skipgr=0 or skipques=0) then skip=-3;
   else if skipgr=-3 and skipleng=0 then skip=-3;
   else if skipleng ge 1 then skip=skipleng;
   else if skipques ge 1 and skipgr ge 1 then skip=skipgr;
   else if skipques ge 1 and skipgr = 0 then skip=-3;
```

```
   else if skipques eq 0 and skipgr ge 1 then skip=-3;
   if skipques=-3 or skipgr=-3 or skipleng=-3 then skip=-3;
end;

if skipj ne -3 then do;
   if (skipl_j=0 and skipgrj=0) or (notskip ge 1 and skipques=0) then skipj=0;
   else if skipl_j eq -3 and (skipgrj=0 or skipques=0) then skip=-3;
   else if skipgrj=-3 and skipl_j=0 then skipj=-3;
   else if skipl_j ge 1 then skipj=skipl_j;
   else if skipques ge 1 and skipgrj ge 1 then skipj=skipgrj;
   else if skipques ge 1 and skipgrj = 0 then skipj=-3;
   else if skipques eq 0 and skipgrj ge 1 then skipj=-3;
   if skip=-3 then skipj=-3;
end;

***Created variable in the previous round;
if cvrpevr3=-5 then cvrpevr3=cvrpevr2;
if cvskevr3=-5 then cvskevr3=cvskevr2;
if cvrpevr3=-5 then cvrpevr3=cvrpevr1;
if cvskevr3=-5 then cvskevr3=cvskevr1;

***Take care of the cases whose"repeated grade" could be double counted;
if oldstrt=1 and repeat>0 and cvrpevr3>0 and pre2857 ne -5 then do;
   do i=1 to 8;
      if preatt[i]=newatt1 and preatt[i+1]=. and (preedate[i]-preadate[i]) gt 13 and preatt[i] ne 95 and (precomp[i]=1 or
            pre2857=preatt[i]) then do;
         repeat=repeat-1;
         if newedat1 le 246 then repeatj=repeatj-1;
      end;
   end;
end;

***-3 the jump rep people;
if jumprep1=1 or jumprep2=1 then do; repeat=-3; repeatj=-3; skip=-3; skipj=-3; end;

***Combine the previous and current information;
if cvrpevr3 ge 0 and repeat ge 0 then numrep=cvrpevr3+repeat;
   else if cvrpevr3 lt 0 then numrep=cvrpevr3;
   else if cvrpevr3 ge 0 and repeat lt 0 then numrep=repeat;
if cvrpevr3 ge 0 and repeatj ge 0 then numrepj=cvrpevr3+repeatj;
   else if cvrpevr3 lt 0 then numrepj=cvrpevr3;
   else if cvrpevr3 ge 0 and repeatj lt 0 then numrepj=repeatj;
if cvskevr3 ge 0 and skip ge 0 then numskip=cvskevr3+skip;
   else if cvskevr3 lt 0 then numskip=cvskevr3;
   else if cvskevr3 ge 0 and skip lt 0 then numskip=skip;
if cvskevr3 ge 0 and skip ge 0 then numskipj=cvskevr3+skipj;
   else if cvskevr3 lt 0 then numskipj=cvskevr3;
   else if cvskevr3 ge 0 and skipj lt 0 then numskipj=skipj;

if s3104=-5 then do; numskip=-5; numrep=-5; numskipj=-5; numrepj=-5; end;

endsas;
```

NUMBER OF SCHOOLS ATTENDED

Variables Created: CV_SCH_ATTEND_EVER
CV_SCH_ATTEND_YR

Variables Used

Name in Program	Question Name on CD	Name in Program	Question Name on CD
Round 1		s826341	YSCH-8263.04.01
curgdr1	YSCH-3500	s826351	YSCH-8263.05.01
id	PUBID	s841611, s841612	YSCH-8416.01.01, .02
numr1	CV_SCH_ATTEND_EVER	s841621, s841622	YSCH-8416.02.01, .02
		s841631, s841632	YSCH-8416.03.01, .02
Round 2		s841641	YSCH-8416.04.01
numr2	CV_SCH_ATTEND_EVER	s841651	YSCH-8416.05.01
hgar2	YSCH-2857	s9589111-s9589114	YSCH-9589.01.01.01-.04
		s9589121	YSCH-9589.01.02.01
Round 3		s9589211-s9589213	YSCH-9589.02.01.01-.03
numr3	CV_SCH_ATTEND_EVER	s9589221	YSCH-9589.02.02.01
hgar3	YSCH-2857	s9589311, s9589312	YSCH-9589.03.01.01, .02
		s9589411	YSCH-9589.04.01.01
Round 4		s9589511	YSCH-9589.05.01.01
hgar4	YSCH-2857	start1m-start5m	NEWSCHOOL_START1.01~M-.05~M
s826311, s826312	YSCH-8263.01.01, .02	start1y-start5y	NEWSCHOOL_START1.01~Y-.05~Y
s826321, s826322	YSCH-8263.02.01, .02	round1-round5	NEWSCHOOL_INTERVIEW.01-.05
s826331, s826332	YSCH-8263.03.01, .02	schid1-schid5	NEWSCHOOL_PUBID.01-.05

This program calculates the number of schools ever attended by the respondent as of the interview date and as of June 30, 2000. The program first counts the number of new regular schools that the youth attended since DLI to the date of survey using the first enrollment date in each new school, then adds this number to the created variable from the previous round to get the total number of regular schools that the youth has ever attended.

Users should note that the variable counts only schools at which the respondent attended grades 7–12. In addition, if a respondent did not have a parent interview in round 1 and the respondent was already in 7th grade or higher in that round, no information was collected on the number of schools attended prior to round 1. These respondents receive a -4 in round 1 and a -3 in all subsequent rounds.

```
array startm start1m start2m start3m start4m start5m ;  array starty start1y start2y start3y start4y start5y ;
array start start1-start5;                               array round round1-round5;
array schid schid1-schid5;                               array dumgrd dumgrd1-dumgrd5;
array again again1-again5;

/*First, create the dummy variable for each school to indicate if the youth attended grade 7 - grade 12 in that school.
dumgrd1-dumgrd5= dummy to indicate whether R attended grades 7-12 in schools 1-5
again1-again5=dummy indicates whether school 1-5 is the same school as one already recorded, to avoid overcounts
startm/y=start date for a given school*/

do i=1 to 5; dumgrd[i]=0; start[i]=-4; end;

if (-3<=s841611<=-1 or -3<=s9589111<=-1 or -3<=s9589112<=-1 or -3<=s9589113<=-1 or -3<=s9589114<=-1) or (-3
        <=s841612<=-1 or -3<=s9589121<=-1) then dumgrd1=-3;
if (-3<=s841621<=-1 or -3<=s9589211<=-1 or -3<=s9589212<=-1 or -3<=s9589213<=-1 ) or -3<=s841622<=-1 then
        dumgrd2=-3;
if (-3<=s841631<=-1 or -3<=s9589311<=-1 or -3<=s9589312<=-1) or -3<=s841632<=-1 then dumgrd3=-3;
if (-3<=s841641<=-1 or -3<=s9589411<=-1) then dumgrd4=-3;
if (-3<=s841651<=-1 or -3<=s9589511<=-1) then dumgrd5=-3;
```

```
if (7<=s841611<=12 or 7<=s9589111<=12 or 7<=s9589112<=12 or 7<=s9589113<=12 or 7<=s9589114<=12) or
        (7<=s841612<=12 or 7<=s9589121<=12) then dumgrd1=1;
if (7<=s841621<=12 or 7<=s9589211<=12 or 7<=s9589212<=12 or 7<=s9589213<=12 ) or 7<=s841622<=12 then
        dumgrd2=1;
if (7<=s841631<=12 or 7<=s9589311<=12 or 7<=s9589312<=12) or 7<=s841632<=12 then dumgrd3=1;
if (7<=s841641<=12 or 7<=s9589411<=12) then dumgrd4=1;
if (7<=s841651<=12 or 7<=s9589511<=12) then dumgrd5=1;

/*create a flag (again 1-5) for schools that are reported more than once for a grade 7-12 in the same round, using pubid
        for the school.  This will be used to avoid double counting schools.*/

do i=1 to 5; again(i)=0; end;
do i=1 to 5;
  if i=5 and schid(i)>0 and dumgrd(i)=1 and dumgrd(i-4)=1 and schid(i)=schid(i-4) then again(i)=1;
  if i>=4 and schid(i)>0 and dumgrd(i)=1 and dumgrd(i-3)=1 and schid(i)=schid(i-3) then again(i)=1;
  if i>=3 and schid(i)>0 and dumgrd(i)=1 and dumgrd(i-2)=1 and schid(i)=schid(i-2) then again(i)=1;
  if i>=2 and schid(i)>0 and dumgrd(i)=1 and dumgrd(i-1)=1 and schid(i)=schid(i-1) then again(i)=1;
end;

array startdm startdm1-startdm5;
do i=1 to 5;
  if starty[i]>0 or starty[i] in (-1,-2,-3) then startdm[i]=1;
  else startdm[i]=0;
end;

/*Next, calculate numjr4 and numr4.*/
 /* Calculate the accumulated month of the start date. */
do i=1 to 5;
  if starty[i]>0 and startm[i]>0 then start[i]=12*(starty[i]-1990)+startm[i];
end;

numjr4=0; numr4=0;

/**** count the new schools for all the cases.*****/
do i=1 to 5;
  if round[i]=4 and again(i)=0 then do;
    if dumgrd[i]=1 then numr4=numr4+1;
    if dumgrd[i]=1 and (0<start[i]<=126 or 0<starty[i]<2000) then numjr4=numjr4+1;
  end;
end;

if -3<=round1<=-1 or -3<=round2<=-1 or -3<=round3<=-1 or -3<=round4<=-1 or -3<=round5<=-1 then do;
  numr4=-3; numjr4=-3;
end;

/**** If the respondent was not interviewed in round 3 and had no number of schools info. in round 1 or 2, or if number
        of schools info. in round 3 is in (0,-4), then count all the schools. That is, for these people, the above count will
        be overwritten.****/
flagprev=0;
if (numr3=-5 and numr2 in (0,-4,-5) and numr1 in (0,-4)) or numr3 in (0,-4,-5) then do;
  numr4=0; numjr4=0; flagprev=1;
  do i=1 to 5;
    if dumgrd[i]=1 and again(i)=0 then numr4=numr4+1;
    if dumgrd[i]=1 and again(i)=0 and (0<start[i]<=126 or 0<starty[i]<2000 or (round[i] in (1,2,3) and again(i)=0)) then
        numjr4=numjr4+1;
  end;
end;
```

```
if dumgrd1=-3 or dumgrd2=-3 or dumgrd3=-3 or dumgrd4=-3 or dumgrd5=-3 then do; numr4=-3; numjr4=-3; end;
if (round1=4 and again1 =0 and dumgrd1=1 and ((start1y=2000 and -4<start1m<0) or -4<start1y<0)) or
   (round2=4 and again2 =0 and dumgrd2=1 and ((start2y=2000 and -4<start2m<0) or -4<start2y<0)) or
   (round3=4 and again3 =0 and dumgrd3=1 and ((start3y=2000 and -4<start3m<0) or -4<start3y<0)) or
   (round4=4 and again4 =0 and dumgrd4=1 and ((start4y=2000 and -4<start4m<0) or -4<start4y<0)) or
   (round5=4 and again5 =0 and dumgrd5=1 and ((start5y=2000 and -4<start5m<0) or -4<start5y<0)) then numjr4=-3;

/*We then add the above two numbers to the corresponding variables from round3, round 2, and/or round 1.*/
if numr3>=0 then do;
   if numjr4>=0 then numj=numjr4+numr3;
   if numr4>=0 then nums=numr4+numr3;
end;
if numr3=-4 and hgar3<7 then do;
   if numjr4>=0 then numj=numjr4;
   if numr4>=0 then nums=numr4;
end;
if numr3=-5 then do;
   if numr2>=0 then do;
      if numjr4>=0 then numj=numjr4+numr2;
      if numr4>=0 then nums=numr4+numr2;
   end;
   if numr1>=0 and numr2<0 then do;
      if numjr4>=0 then numj=numjr4+numr1;
      if numr4>=0 then nums=numr4+numr1;
   end;

/*For the youth who didn't answer the number of schools in round 1, we distinguish them into three cases 1)If the 'grade
         currently attended' in round 1 is <=7 and the 'highest grade attended' in round 3 >= 7, we use the number from
         round 3 as the total number. 2)If the 'grade currently attended' in round 1 >7, we use -3 for the total number.
         3)If both the 'grade currently attended' in round 1 and the 'highest grade attended' in round 3 are <7, we use -4
         for the total number.*/
   if numr1 in (0,-4) then do;
      if curgdr1<=7 then do; numj=numjr4; nums=numr4; end;
      if curgdr1>7 or numr1 in (-1,-2,-3) then do; numj=-3; nums=-3; end;
      if curgdr1<7 and hgar4<7 then do; numj=-4; nums=-4; end;
   end;
end;

if numr1 in (-4,0) and curgdr1>7 then do; nums=-4; numj=-4; end;
if -3<=numr1<=-1 or -3<=numr2<=-1 or numr3=-3 or numjr4=-3 then numj=-3;
if -3<=numr1<=-1 or -3<=numr2<=-1 or numr3=-3 or numr4=-3 then nums=-3;
if hgar4<7 then do; numj=-4; nums=-4; end;

if s826311=-5 then do; numjr4=-5; numr4=-5; numj=-5; nums=-5; end;

/*hand edits:*/
if id=1411763 then do; nums=-3; numj=-3; end;        if id=1470481 then do; nums=-3; numj=-3; end;
if id=1046701 then do; nums=-3; numj=-3; end;        if id=1402382 then do; nums=-3; numj=-3; end;

/*these Rs were filled in with -4 because their HGA on the single item question was <7 but they report attending grades
         7-12 in loops in prior rounds.  None report new schools in the current round so their CV equals that of the
         prior round;*/
if id in (1024411, 1044381, 1083491, 1105751, 1118571, 1398031, 1454112, 1855201) then do; nums=numr3;
         numj=numr3; end;

endsas;
```

TOTAL FRACTION OF CREDITS EARNED TOWARDS BACHELORS/ASSOCIATE DEGREE

Variables Created: CV_BA_CREDITS.01–.05
CV_ASSOC_CREDITS.01–.05

Variables Used

Name in Program	Question Name on CD	Name in Program	Question Name on CD
Round 1		**Round 4**	
id	PUBID	e16200	YSCH-16200.01
r1_asct	CV_ASSOC_CREDITS	e265211-e265214	YSCH-26521.01-.04
r1_bact	CV_BA_CREDITS	xcred1-xcred4	YSCH-26827.01-.04
		apcred1-apcred4	YSCH-26878.01-.04
Round 2		hscred1-hscred4	YSCH-26929.01-.04
r2pubid1-r2pubid6	NEWSCHOOL_PUBID.01-.06	e273371-e273374	YSCH-27337.01-.04
r2_asc1-r2_asc5	CV_ASSOC_CREDITS.01-.05	e273881-e273884	YSCH-27388.01-.04
r2_bac1-r2_bac5	CV_BA_CREDITS.01-.05	e228001-e228004	YSCH-22800.01-.04
		e230001-e230003	YSCH-23000.01-.03
Round 3		r4pubid1-r4pubid5	NEWSCHOOL_PUBID.01-.05
r3pubid1-r3pubid7	NEWSCHOOL_PUBID.01-.07		
r3_bac1-r3_bac5	CV_BA_CREDITS.01-.05		

Codes for Created Variable

These variables are created on a continuous scale. They have 2 implied decimal places.

This program calculates the fraction of credits the respondent has completed toward an associate's degree or a bachelor's degree.

```
/* e23000(1-3) accounts for incorrect tabulation of Round3 credits earned. */
array e23000 (i) e230001-e230004;          array e22800 (i) e228001-e228004;
array xcred (i) xcred1-xcred4;             array apcred (i) apcred1-apcred4;
array hscred (i) hscred1-hscred4;          array tcred (i) tcred1-tcred4;
array tascred (i) tascred1-tascred4;       array tbacred (i) tbacred1-tbacred4;
array asfrac (i) asfrac1-asfrac4; /* created variable for associate credits */
array bafrac (i) bafrac1-bafrac4; /* created variable for bachelor credits */
array e27337 (i) e273371-e273374;          array e27388 (i) e273881-e273884;
array r3tbac (i) r3tbac1-r3tbac5;          array r2tbac (i) r2tbac1-r2tbac5;
array r3tasc (i) r3tasc1-r3tasc5;          array r2tasc (i) r2tasc1-r2tasc5;
array r4tbac (i) r4tbac1-r4tbac4;          array r4tasc (i) r4tasc1-r4tasc4;
array e26521 (i) e265211-e265214;          array e23200 (i) e232001-e232004;

/* Initialize both created variables. */
do i=1 to 4; asfrac=0; bafrac=0; end;

/* If a respondent completed at least one term, CAPI verifies the total credits for each school at YSCH-22800.  Due to a
      design error, the check question YSCH-22800 does not include R2 college credits even when the PUBID's from
      both rounds match. However, if a respondent see this incorrect count and tells us it is correct, we will use it.
      Respondents who have yet to complete a term at a given school will not be asked YSCH-22800. The only
      credits possible for these people are incoming credits at YSCH-26521, or the dli Round c.v. for a matching
      school. */
/* There are 3 ways to get credits in this program, which are all exclusive (except for 1 case mentioned below).  If the
      respondent completed any terms, we use YSCH-22800 and/or YSCH-23000. If the respondent has not
      completed any terms and reported a new school, then we use the incoming credits (YSCH-26521). If the
      respondent has not completed any terms and reported a dli school, then we use the total credit variable from the
      dli round (through the PUBID match variable described below).  */
array r4cred (i) r4cred1-r4cred4;
do i=1 to 4;
```

```
  if e22800>-4 then do; tcred=e22800; r4cred=1; end;
  if e23000>-4 then do; tcred=e23000; r4cred=1; end;
end;

/* Check for people who attended college but were skipped out of YSCH-22800 and YSCH-23000 (because no
        completed terms). */
array noterm (i) noterm1-noterm4;
do i=1 to 4;
  if e27337 in (1,3) and e22800=-4 and e23000=-4 then noterm=1;
end;

/* For people who did not complete any terms, the only data available is the incoming credits count and the created
        variable from their dli round. These two pieces of data are exclusive. */
do i=1 to 4;
  if noterm=1 and (r3tbac>0 or r3tasc>0) then do; back3=1; end;
  if noterm=1 and (r2tbac>0 or r2tasc>0) and r3int=-5 then do; back2=1; end;
end;

/* The flag variable checks for inconsistency of incoming credits answers. When reporting a new college, we ask the
        respondent for total incoming credits (YSCH-26521) and then to break them up into transfer credits, college
        credits earned in high school, & AP credits. When the sum of the three components doesn't equal the answer in
        YSCH-26521, flag=1. */
array flag (i) flag1-flag4;
do i=1 to 4;
  if E26521=>0 and XCRED=>0 and HSCRED=>0 and APCRED=>0 then do;
    if E26521 ne XCRED+HSCRED+APCRED then do; flag=1; end;
  end;
end;

/* The following are people who reported a new school and have no completed terms (and thus no answer to YSCH-
        22800). */
array incred (i) incred1-incred4;
do i=1 to 4;
  if e26521>-4 and e22800=-4 and e23000=-4 then do; tcred=e26521; incred=1; end;
end;

/* The last data source for credits are previous round data. Begin by matching Round 4 colleges to dli colleges using the
        pubid's. Note that this is only used when there are no completed terms in Round 4 (and thus no answer to
        YSCH-22800) and this school is a dli school (thus no answer to YSCH-26521). The pubid's omitted from the
        array below have no colleges. */
array r34m1 (i) r34m11-r34m14;          array r34m2 (i) r34m21-r34m24;
array r34m3 (i) r34m31-r34m34;          array r34m4 (i) r34m41-r34m44;
array r34m5 (i) r34m51-r34m54;          array r24m1 (i) r24m11-r24m14;
array r24m2 (i) r24m21-r24m24;          array r24m3 (i) r24m31-r24m34;
array r24m4 (i) r24m41-r24m44;          array r24m5 (i) r24m51-r24m54;
array r2pub (i) r2pubid1-r2pubid5;      array r3pub (i) r3pubid1-r3pubid4;
array r4pub (i) r4pubid1-r4pubid4;

do i=1 to 4;
  r34m1=0; r34m2=0; r34m3=0; r34m4=0;
  r24m1=0; r24m2=0; r24m3=0; r24m4=0; r24m5=0;
end;

/* 10-22-01 Define r34m14 as the dummy variable that equals one when the first college on the Round3 school roster
        and the fourth college on the Round4 roster have the same PUBID. Define r24m14 as the dummy variable that
        equals one when the first college on the Round2 school roster and the fourth college on the Round4 roster have
        the same PUBID. The "24" match variables require that R missed the Round 3 interview. Any college from
```

```
        Round1 is represented in the "first" position in the Round 2 created variables; i.e. ASFRAC1 (Round 2) is a
        school from Round 1. */

do over r4pub;              /*If any PUBID from the first position in Round 2/3 matches any PUBID in Round 4*/
  if r3pubid1>0 and r4pub>0 and r3int>0 then do;
    if r3pubid1=r4pub then do; r34m1=1; end;
  end;
  if r2pubid1>0 and r4pub>0 and r3int=-5 and r2int>0 then do;
    if r2pubid1=r4pub then do; r24m1=1; end;
  end;
end;
do over r4pub;              /*If any PUBID from the second position in Round 2/3 matches any PUBID in Round 4*/
  if r3pubid2>0 and r4pub>0 and r3int>0 then do;
    if r3pubid2=r4pub then do; r34m2=1; end;
  end;
  if r2pubid2>0 and r4pub>0 and r3int=-5 and r2int>0 then do;
    if r2pubid2=r4pub then do; r24m2=1; end;
  end;
end;
do over r4pub;              /*If any PUBID from the third position in Round 2/3 matches any PUBID in Round 4*/
  if r3pubid3>0 and r4pub>0 and r3int>0 then do;
    if r3pubid3=r4pub then do; r34m3=1; end;
  end;
  if r2pubid3>0 and r4pub>0 and r3int=-5 and r2int>0 then do;
    if r2pubid3=r4pub then do; r24m3=1; end;
  end;
end;
do over r4pub;              /*If any PUBID from the fourth position in Round 2/3 matches any PUBID in Round 4*/
  if r3pubid4>0 and r4pub>0 and r3int>0 then do;
    if r3pubid4=r4pub then do; r34m4=1; end;
  end;
  if r2pubid4>0 and r4pub>0 and r3int=-5 and r2int>0 then do;
    if r2pubid4=r4pub then do; r24m4=1; end;
  end;
end;
do over r4pub;              /*If any PUBID from the fifth position in Round 2/3 matches any PUBID in Round 4*/
  if r2pubid5>0 and r4pub>0 and r3int=-5 and r2int>0 then do;
    if r2pubid5=r4pub then do; r24m5=1; end;
  end;
end;

/* Create a dummy that equals one if any of the "match" variables above equal one. This is just a shortcut, so we don't
        have to keep writing "or match11=1 or match12=1..." anytime we want to look at people with identical Round2
        and Round3 colleges. */
do i=1 to 4;
  if r34m1=1 or r34m2=1 or r34m3=1 or r34m4=1 or r24m1=1 or r24m2=1 or r24m3=1 or r24m4=1 or r24m5=1 then
        do; same=1; end;
end;

/* We will accept dli credits without a matching degree working towards answer from current and dli round. For example,
        if a respondent reports working toward a associate's in his/her dli round and then reports working toward a
        bachelor's in the current round, we will still use the total dli credits in the current round c.v., as long as the
        pubid's match (i.e., it is the same school). */
array dlicred (i) dlicred1-dlicred4;
do i=1 to 4;
  if r2tasc1=>0 and r24m1=1 and e22800=-4 then do; tcred=r2tasc1; dlicred=1; end;
  if r2tbac1=>0 and r24m1=1 and e22800=-4 then do; tcred=r2tbac1; dlicred=1; end;
```

```
   if r2tasc2=>0 and r24m2=1 and e22800=-4 then do; tcred=r2tasc2; dlicred=1; end;
   if r2tbac2=>0 and r24m2=1 and e22800=-4 then do; tcred=r2tbac2; dlicred=1; end;
   if r2tasc3=>0 and r24m3=1 and e22800=-4 then do; tcred=r2tasc3; dlicred=1; end;
   if r2tbac3=>0 and r24m3=1 and e22800=-4 then do; tcred=r2tbac3; dlicred=1; end;
   if r2tasc4=>0 and r24m4=1 and e22800=-4 then do; tcred=r2tasc4; dlicred=1; end;
   if r2tbac4=>0 and r24m4=1 and e22800=-4 then do; tcred=r2tbac4; dlicred=1; end;
   if r2tasc5=>0 and r24m5=1 and e22800=-4 then do; tcred=r2tasc5; dlicred=1; end;
   if r2tbac5=>0 and r24m5=1 and e22800=-4 then do; tcred=r2tbac5; dlicred=1; end;
end;

do i=1 to 4;
   if r3tasc1=>0 and r34m1=1 and e22800=-4 then do; tcred=r3tasc1; dlicred=1; end;
   if r3tbac1=>0 and r34m1=1 and e22800=-4 then do; tcred=r3tbac1; dlicred=1; end;
   if r3tasc2=>0 and r34m2=1 and e22800=-4 then do; tcred=r3tasc2; dlicred=1; end;
   if r3tbac2=>0 and r34m2=1 and e22800=-4 then do; tcred=r3tbac2; dlicred=1; end;
   if r3tasc3=>0 and r34m3=1 and e22800=-4 then do; tcred=r3tasc3; dlicred=1; end;
   if r3tbac3=>0 and r34m3=1 and e22800=-4 then do; tcred=r3tbac3; dlicred=1; end;
   if r3tasc4=>0 and r34m4=1 and e22800=-4 then do; tcred=r3tasc4; dlicred=1; end;
   if r3tbac4=>0 and r34m4=1 and e22800=-4 then do; tcred=r3tbac4; dlicred=1; end;
end;

/* Associates Degree */
do i=1 to 4;
   if e27337=1 and tcred=>0 then do; r4tasc=tcred; end;
   if -4<e27388<0 and e27337=1 then do; asfrac=e27388; end;
   if -4<tcred<0 and e27337=1 then do; asfrac=tcred; end;
   if r4tasc=>0 and e27388>0 then do; asfrac=(r4tasc/e27388)*100; end;
   asfrac=round(asfrac,1);
   /* respondents who say they have graduated */ if e23200=1 and e27337=1 then asfrac=100;
end;

/* Bachelors Degree */
do i=1 to 4;
   if e27337=3 and tcred=>0 then do; r4tbac=tcred; end;
   if -4<e27388<0 and e27337=3 then do; bafrac=e27388; end;
   if -4<tcred<0 and e27337=3 then do; bafrac=tcred; end;
   if r4tbac=>0 and e27388>0 then do; bafrac=(r4tbac/e27388)*100; end;
   bafrac=round(bafrac,1);
   /* respondents who say they have graduated */ if e23200=1 and e27337=3 then bafrac=100;
end;

/* Non-interview & invalid data cases */
do i=1 to 4;
   if e16200=-5 then do; bafrac=-5; asfrac=-5; end;
end;

do i=1 to 4;
   if asfrac in (-2,-1) then do; asfrac=-3; end;
   if bafrac in (-2,-1) then do; bafrac=-3; end;
end;

/* These people we calculate 100 percent credits earned but say they haven't graduated. After looking at their college
          credits answers, we decided to assign them -3's in their respective c.v.'s. */
if id in (1079191, 1483701) then asfrac1=-3;
if id in (1636161, 7350131) then asfrac2=-3;
if id=4430021 then bafrac1=-3;
```

```
/* 15 people have either amiss or bmiss. Some of them have valid dli info but switched the degree working towards, some
        have zeros in credits earned, and some have zeros in credits needed to graduate. */
do i=1 to 4;
  if asfrac>100 or bafrac>100 then over=1;
  if e27337=1 and asfrac=. then amiss=1;
  if e27337=3 and bafrac=. then bmiss=1;
end;

/* There are 28 people with YSCH-27388 (# of credits needed to graduate) equal to zero. Information was not available
        from previous rounds.  */
do i=1 to 4;
  if e23200=1 and e27337=1 and 0<asfrac<100 then acheck=1;
  if e23200=1 and e27337=3 and 0<bafrac<100 then bcheck=1;
end;

do over e27337;
  if e27337=-4 then do; asfrac=-4; bafrac=-4; end;
end;

do over e27337;
  if e27337=1 then do; bafrac=-4; end;
end;

do over e27337;
  if e27337=3 then do; asfrac=-4; end;
end;

endsas;
```

YOUTH'S MATH PIAT SCORE

Variables Created: CV_PIAT_STANDARD_SCORE
CV_PIAT_PERCENTILE_SCORE

Variables Used

Name in Program	Question Name on CD
bdate_d, m, y	KEY!BDATE_D, _M, _Y (round 1)
intdated, m, y	CV_INTERVIEW_DATE~D, ~M, ~Y
piat_r00	PIAT_RAW_SCORE

This program calculates the respondent's age utilizing the interview date information and the respondent's date of birth information. It then uses the age data and the raw PIAT score to create a standard PIAT score and percentile rank for each respondent.

```
/***************interview date section: calculate the interview date in continuous months format******************/
if intdated  ge  0 and intdatem  ge  0 then int_cmon=((intdatey-1990)*12)+intdatem;
   else if intdatey eq -5 then int_cmon=-5;
if int_cmon=. then do;
    put 'PROBLEM: Unexpected value for interview date variable ' id= intdated= intdatem= intdatey=;
end;

/****************respondent age section****************/
if bdate_y lt 1980 or bdate_y gt 1984 then do;
   put 'PROBLEM: Unexpected value for birthdate year ' id= bdate_d= bdate_m= bdate_y=;
end;

/**calculate the youth's age in continuous months**/
if int_cmon ne . then do;
   age_cmon=((intdatey-bdate_y)*12)+(intdatem-bdate_m);
   if intdated-bdate_d<0 then age_cmon=age_cmon-1;
end;


/********************piat standard score section************/
if PIATR00 lt 0 then piat_s00=PIATR00;
else if PIATR00 ge 0 then do;

/* At this point the program loops through each three-month age range and assigns appropriate standard scores based on
the respondent's raw score.  This code is quite lengthy and is not reproduced here; instead, we include a table that
provides the standard score and percentile rank associated with each raw score in each age range.  Users who need the
precise programming code should contact NLS User Services. */

********************piat percentile rank section************;

if piat_s00 lt 0 then piat_p00=piat_s00;
else if piat_s00 ge 0 then do;

/* At this point the program assigns percentile scores based on the standard scores.  These are also shown in the table on
the following page. */

endsas;
```

Standard Score	Percentile Rank	Raw Score by Age Range (Years-Months)						Standard Score	Percentile Rank	Raw Score by Age Range (Years-Months)					
		15-9 to 15-11	16-0 to 16-2	16-3 to 16-5	16-6 to 16-8	16-9 to 16-11	17-0 to 17-2			15-9 to 15-11	16-0 to 16-2	16-3 to 16-5	16-6 to 16-8	16-9 to 16-11	17-0 to 17-2
55	0	≤26	≤27	≤27	≤28	≤28	≤28	99	47	85	85	86	86	86	87
56	0	27	28	28	29	29	29	100	50	86	86	87	87	87	88
57	0	28	29	29			30	101	53	87	87	88	88	88	
58	0	29			30	30	31	102	55	88	88				89
59	0		30	30	31	31	32	103	58			89	89	89	90
60	0	30	31	31	32	32	33	104	61	89	89	90	90		
61	0	31	32	32	33	33	34	105	63	90	90			91	91
62	1	32	33	33	34	34	36	106	66			91	91	92	92
63	1	33	34	34	36	36	37	107	68	91	91	92	92		
64	1	34	36	37	37	38	38	108	70	92	92			93	93
65	1	36	37	38	38	39	39	109	73		93	93	93		
66	1	38	38	39	39	40	40	110	75	93			94	94	94
67	1	39	39	40	40	41	41	111	77		94	94			
68	2	40	40	41	41	43	43	112	79	94					
69	2	41	41	43	43	44	44	113	81				95	95	95
70	2	43	43	44	45	45	45	114	82		95	95			
71	3	44	45	45	47	47	48	115	84	95					96
72	3	46	47	47	49	49	50	116	86			96	96	96	
73	4	48	49	49	50	50	52	117	87	96	96				
74	4	49	50	50	52	52	53	118	88				97	97	97
75	5	51	52	52	53	53	54	119	90		97	97			
76	5	53	53	54	54	54	56	120	91	97					98
77	6	54	54	56	56	56	58	121	92			98	98	98	
78	7	57	56	58	58	58	59	122	93	98	98				
79	8	58	58	59	59	60	61	123	94						
80	9	59	59	61	61	62	62	124	95						
81	10	61	61	62	62	64	64	125	95						
82	12	62	62	64	64	66	66	126	96						
83	13	64	64	66	66	68	68	127	96						
84	14	66	67	68	68	69	69	128	97						
85	16	68	69	69	69	70	70	129	97						
86	18	69	70	70	71	71	71	130	98					99	99
87	19	70	71	71	72	72	72	131	98			99	99		
88	21	71	72	72	74	74	74	132	98	99	99				
89	23	72	74	74	75	75	75	133	99						
90	25	74	75	75	76	76	76	134	99						
91	27	75	76	76	77	77	77	135	99						
92	30	76	77	77	78	78	79	136	99						
93	32	78	78	78	79	79	81	137	99						
94	34	79	79	79	81	81	82	138	99						
95	37	81	81	81	82	82	83	139	100						
96	39	82	82	82	83	83	84	140	100				100	100	100
97	42	83	83	83	84	84	85	141	100		100	100			
98	45	84	84	85	85	85	86	142	100	100					

Note: This table of *PIAT* scores is based on the information provided by the testing company. Users interested in more information about the *PIAT-Math Assessment* may wish to consult the following reference:

Markwardt, Frederick C. Jr. *Peabody Individual Achievement Test-Revised*. Circle Pines, Minn.: American Guidance Service, Inc., 1989.

QED DATA: SCHOOL SIZE AND STUDENT-TEACHER RATIO

Variables Created: CV_SCHOOL_SIZE
CV_STUDENT_TEACHER_RATIO

Variables Used

Name in Program	Question Name on CD	Name in Program	Question Name on CD
pubid	PUBID	starm11-starm51	NEWSCHOOL_START1.01~M-.05~M
e2857	YSCH-2857	stary11-stary51	NEWSCHOOL_START1.01~Y-.05~Y
e21625	YSCH-21625	starm12, stary12	NEWSCHOOL_START2.01~M, ~Y
pin1-pin5	NEWSCHOOL_ID.01-.05	starm22, stary22	NEWSCHOOL_START2.02~M, ~Y
age	SYMBOL!KEY!AGE	starm32, stary32	NEWSCHOOL_START2.03~M, ~Y
race	SYMBOL!KEY!RACE	stopm11-stopm51	NEWSCHOOL_STOP1.01~M-.05~M
gender	SYMBOL!KEY!SEX	stopy11-stopy51	NEWSCHOOL_STOP1.01~Y-.05~Y
		stopm12, stopy12	NEWSCHOOL_STOP2.01~M, ~Y
		stopm22, stopy22	NEWSCHOOL_STOP2.02~M, ~Y
		stopm32, stopy32	NEWSCHOOL_STOP2.03~M, ~Y
		type1-type5	NEWSCHOOL_SCHCODE.01-.05

Codes for Created Variable

School Size	Student-teacher ratio
1 = <100	1 = <14
2 = 100-299	2 = 14-17
3 = 300-499	3 = 18-21
4 = 500-749	4 = 22+
5 = 750-999	
6 = 1000+	

This program merges school identification information from the NLSY97 data with data from the "National Education Database," provided under copyright by Quality Education Data (QED), Inc. It then creates two variables that provide information about the respondent's most recent K-12 school.

/* Before creating the QED variables, survey staff used dates of enrollment data to identify the respondent's current or most recent school and then merged that information with the QED data. The process for identifying the most recent school is the same as that used in the creation of the CV_SCHOOL_TYPE variable, which appears earlier in this section. This information is excluded here due to length and confidentiality restrictions. Researchers needing more information about this process should contact NLS User Services. */

```
if e21625=999 then e21625=9;
schsize=-4;
studteac=-4;

/* Now, create schsize and studteac while checking that
      the grade reported in the QED coincides with that
      being reported in the NLSY97  */

if gradspan='"2"' then gradspan='"3"';
if gradspan='"a"' then gradspan='"2"';
if gradspan='"b"' or gradspan='"c"' or gradspan='"d"' or
      gradspan='"e"' or gradspan='"*"' then
      gradspan='"0"';

/* Redefine each gradspan to gradspn to eliminate
      character value problems */
if gradspan='"0"' then gradspn=0;
if gradspan='"1"' then gradspn=1;
if gradspan='"2"' then gradspn=2;
if gradspan='"3"' then gradspn=3;
if gradspan='"4"' then gradspn=4;
if gradspan='"5"' then gradspn=5;
if gradspan='"6"' then gradspn=6;
if gradspan='"7"' then gradspn=7;
if gradspan='"8"' then gradspn=8;
if gradspan='"9"' then gradspn=9;

/* For respondents with e2857>12 but have a matching
      PIN number we will use the school with the
      matching pin instead of the college. Even though
      the college is the most recent school in these cases,
      the QED c.v.'s refer to the most recent high school
      (or middle, elementary, etc.).  */
if e2857>12 and pin>100 and typesch=3 then do;
  e2857=12; hgaflag=1;
end;
```

```
 dummy1=0;        dummy4=0;        dummy5=0;
 dummy6=0;        dummy7=0;        dummy8=0;
 dummy9=0;        dummy10=0;
if e2857 ge 1 and e2857 le 3 then do;
 dummy1=1;        dummy4=1;        dummy5=1;
 dummy6=0;        dummy7=0;        dummy8=0;
 dummy9=0;        dummy10=0;
end;
if e2857 ge 4 and e2857 le 6 then do;
 dummy1=1;        dummy4=1;        dummy5=1;
 dummy6=1;        dummy7=0;        dummy8=0;
 dummy9=0;        dummy10=0;
end;
if e2857 ge 7 and e2857 le 8 then do;
 dummy1=1;        dummy4=0;        dummy5=1;
 dummy6=1;        dummy7=1;        dummy8=1;
 dummy9=0;        dummy10=0;
end;
if e2857 eq 9 then do;
 dummy1=1;        dummy4=0;        dummy5=0;
 dummy6=0;        dummy7=1;        dummy8=1;
 dummy9=1;        dummy10=0;
end;
if e2857 ge 10 and e2857 le 12 then do;
 dummy1=1;        dummy4=0;        dummy5=0;
 dummy6=0;        dummy7=0;        dummy8=1;
 dummy9=1;        dummy10=1;
end;

correct=0;
if gradspn=1 and dummy1=1 then correct=1;
if gradspn=4 and dummy4=1 then correct=1;
if gradspn=5 and dummy5=1 then correct=1;
if gradspn=6 and dummy6=1 then correct=1;
if gradspn=7 and dummy7=1 then correct=1;
if gradspn=8 and dummy8=1 then correct=1;
if gradspn=9 and dummy9=1 then correct=1;
if gradspn=2 and dummy10=1 then correct=1;

/* Rename character student range variable to numeric
      variable. */
if studrang='"0"' then stdrang=0;
if studrang='"1"' then stdrang=1;
if studrang='"2"' then stdrang=2;
if studrang='"3"' then stdrang=3;
if studrang='"4"' then stdrang=4;
if studrang='"5"' then stdrang=5;
if studrang='"6"' then stdrang=6;
if studrang='"7"' then stdrang=7;
if studrang='"8"' then stdrang=8;
if studrang='"9"' then stdrang=9;
if studrang='"."' then stdrang=-3;

/* If the grades reported in the QED and the NLSY97 do
      not coincide, we make both created variables equal
      to -3; otherwise, we proceed to create them
      according to the definition provided at the top of
      the program */
if studrang='"0"' or studrang='"1"' then schsize=1;
if studrang='"2"' then schsize=2;
if studrang='"3"' then schsize=3;
if studrang='"4"' then schsize=4;
if studrang='"5"' then schsize=5;
if studrang='"6"' or studrang='"7"' or studrang='"8"' or
      studrang='"9"' then schsize=6;
if studrang='"."' then schsize=-3;

if nostud gt 0 and noteach gt 0 then
      studtea1=nostud/noteach;
if studtea1 lt 14 then studteac=1;
if studtea1 ge 14 and studtea1 lt 18 then studteac=2;
if studtea1 ge 18 and studtea1 lt 22 then studteac=3;
if studtea1 ge 22 then studteac=4;
if studtea1 eq . then studteac=-3;

if correct=0 then schsize=-3;
if correct=0 then studteac=-3;

if e2857=-5 then schsize=-5;
if e2857=-5 then studteac=-5;

/* We also make sure that there are not five or less schools
      in each cell determined by the region, type of
      school, control, school size and the student-teacher
      ratio to avoid any possibilities of identifying the
      schools. We first run the tabulations, then assign -3
      to the created variables for schools that fall in cells of
      five or less schools. */
index=census||typesch||e21625||schsize||studteac;
proc freq;
 tables census*typesch*e21625*schsize*studteac / out=a;
 proc freq data=A noprint;
 tables count;
data A;
set a;
index=census||typesch||e21625||schsize||studteac;

if count le 5 then schsize=-3;
if count le 5 then studteac=-3;
if e2857=-5 then do; studteac=-5; schsize=-5; end;

/*If a respondent reports no schools in the Round 3 roster
      or the only schools in the R3 roster are 2 or 4 year
      universities */
if pin=-4 then schsize=-4;
if pin=-4 then studteac=-4;

endsas;
```

TRAINING: RECEIPT OF CERTIFICATE OR VOCATIONAL LICENSE

Variables Created: CV_TRN_CERT
CV_TRN_CERT_DATE~M, ~Y

Variables Used

Name in Program	Question Name on CD	Name in Program	Question Name on CD
t6400d1-t6400d3	YTRN-640.01~D-.03~D	t1500m1-t1500m7	YTRN-1500.01~M-.07~M
t6400m1-t6400m3	YTRN-640.01~M-.03~M	t1500y1-t1500y7	YTRN-1500.01~Y-.07~Y
t6400y1-t6400y3	YTRN-640.01~Y-.03~Y	trncrt97-99	CV_TRN_CERT (rounds 1-3)
t690001-t690007	YTRN-6900.01-.07	trncdy97, 98	internal created variable for day of receipt
t700001-t700004	YTRN-7000.01-.04	trncmo97-99	CV_TRN_CERT_DATE (rounds 1-3)
t1500d1-t1500d7	YTRN-1500.01~D-.07~D	trncy97-99	CV_TRN_CERT_DATE (rounds 1-3)

Codes for Created Variable

Ever Received Vocational/Completion Certificate or State License (TRNCERT)
0 Condition Does Not Apply (no)--R took a training, but didn't earn cert.
1 Condition Applies (yes)

This program creates a variable that reports whether the respondent has ever received a certificate or license other than a GED through a training program. If so, a pair of variables provide the month and year that the most recent certificate or license was obtained.

```
TRNCERT=.; /*Ever received vocational/completion certificate or state license*/
TRNCMTH=.; /*Month of most recent certificate/license--based on date of completion of most recent cert/lic granting
        training program*/
TRNCYR=.;  /*Year of most recent certificate/license--based on date of completion of most recent cert/lic granting
        training program*/
reccert=0; /* a counting variable for use in programming to check do loops*/

/*Defining arrays*/
array t69000{5} t69001 - t69005;          array t70000{5} t70001 - t70005;
array t1500m{5} t1500m1 - t1500m5;        array t1500y{5} t1500y1 - t1500y5;
array t6400m{5} t6400m1 - t6400m5;        array t6400y{5} t6400y1 - t6400y5;
array trncrt{3} trncrt97 - trncrt99;      array trncmo{3} trncmo97 - trncmo99;
array trncy{3} trncy97 - trncy99;

do i=1 to 5;
  if reccert=0 and t69000{i}=1 and 1<=t70000{i}<=3 and t6400y{i}>1 then do;
     TRNCERT=1; TRNCYR=t6400y{i}; TRNCMTH=t6400m{i}; reccert=1;
  end;
  if reccert=0 and t69000{i}=1 and 1<=t70000{i}<=3 and t1500y{i}>1 then do;
     TRNCERT=1; TRNCYR=t1500y{i}; TRNCMTH=t1500m{i}; reccert=1;
  end;
  if reccert>=1 and t69000{i}=1 and 1<=t70000{i}<=3 and t1500y{i}=TRNCYR and t1500m{i}>TRNCMTH then do;
     TRNCMTH=t1500m{i}; reccert=reccert+1;
  end;
  if reccert>=1 and t69000{i}=1 and 1<=t70000{i}<=3 and t6400y{i}=TRNCYR and t6400m{i}>TRNCMTH then do;
     TRNCMTH=t6400m{i}; reccert=reccert+1;
  end;
  if reccert>=1 and t69000{i}=1 and 1<=t70000{i}<=3 and t1500y{i}>TRNCYR then do;
    TRNCYR=t1500y{i}; TRNCMTH=t1500m{i}; reccert=reccert+1;
  end;
  if reccert>=1 and t69000{i}=1 and 1<=t70000{i}<=3 and t6400y{i}>TRNCYR then do;
    TRNCYR=t6400y{i}; TRNCMTH=t6400m{i}; reccert=reccert+1;
  end;
```

```
  if TRNCERT^=1 then do;
    if t69000{i}=0 then TRNCERT=0;
    if t69000{i}=1 and t70000{i}>3 then TRNCERT=0;
    if t69000{i}=1 and t70000{i}<1 then TRNCERT=0;
  end;
end;

/*code non-interviews*/
if t69001=-5 then do;
  TRNCERT=-5; TRNCYR=-5;  TRNCMTH=-5;
end;

/*This part compares values for current round to created vars for previous round(s) to obtain the most recent date of
          cert/lic receipt*/
do i=1 to 3;
  if trncert=. and trncrt{i}=1 then do;
    TRNCERT=TRNCRT{i}; TRNCYR=TRNCY{i}; TRNCMTH=TRNCMO{i};
  end;
  if trncert=0 and trncrt{i}=1 then do;
    TRNCERT=TRNCRT{i}; TRNCYR=TRNCY{i}; TRNCMTH=TRNCMO{i};
  end;
  if TRNCERT=1 and TRNCRT{i}=1 and TRNCY{i}>TRNCYR then do;
    TRNCYR=TRNCY{i}; TRNCMTH=TRNCMO{i};
  end;
  if TRNCERT=1 and TRNCRT{i}=1 and TRNCY{i}=TRNCYR and TRNCMO{i}>TRNCMTH then do;
    TRNCMTH=TRNCMO{i};
  end;
  if TRNCERT=. and TRNCRT{i}=0 then do;
    TRNCERT=TRNCRT{i}; TRNCYR=TRNCY{i}; TRNCMTH=TRNCMO{i};
  end;
end;

if TRNCYR=-2 or TRNCYR=-1 then TRNCYR=-3;
if TRNCMTH=-2 or TRNCMTH=-1 then TRNCMTH=-3;
if TRNCERT=. and t69001=-4 then TRNCERT=-4;
if TRNCERT=. and t69001=-5 then TRNCERT=-5;
if TRNCERT=. and t69001=-2 then TRNCERT=-3;
if TRNCERT=. and t69001=-1 then TRNCERT=-3;
if TRNCERT=. and t69001=-3 then TRNCERT=-3;
if TRNCERT=. then TRNCERT=0;
if TRNCERT=1 and TRNCMTH<1 then TRNCMTH=-3;
if TRNCERT=1 and TRNCYR<1 then TRNCYR=-3;

/*if R took training but did not receive training cert then the date variables= valid skip  */
if TRNCERT=0 then TRNCMTH=-4;
if TRNCERT=0 then TRNCYR=-4;

/*if R received training cert but there is no date, then date variables = invalid skip*/
if TRNCERT=-3 then TRNCMTH=-3;
if TRNCERT=-3 then TRNCYR=-3;

/*valid skip*/  if TRNCERT=-4 then TRNCMTH=-4;
if TRNCERT=-4 then TRNCYR=-4;
/*noninterview*/  if TRNCERT=-5 then TRNCMTH=-5;
if TRNCERT=-5 then TRNCYR=-5;

endsas;
```

NLSY97 Appendix 2:

Employment Variable Creation

INTRODUCTION

Collection of Employment Data in Round 4. The employment sections of the NLSY97 questionnaire are somewhat complex. Before beginning analysis, researchers must understand the structure of each round's questionnaire, particularly the way in which jobs are classified as employee, freelance, or self-employment. It is important to note that this classification depends in part on the survey round and the respondent's age. In rounds 1 and 2, employee jobs were recorded in the first part of the YEMP section, administered only to respondents age 14 or older as of the interview date. The second part of the YEMP section collected information about freelance jobs of respondents age 14 and older and all jobs of respondents age 12 or 13 (the implicit assumption being that respondents younger than 14 are not likely to hold employee jobs). If the respondent was at least 16 years old and made at least $200/week in a freelance job, the job was classified as self-employment and an extra series of questions was asked during the freelance section.

In round 3, all respondents were at least age 14 by the interview date, so the age restriction for employee jobs was no longer necessary. The structure of the section remained largely the same, with a division between employee and freelance jobs. Self-employment was classified in the same way as in the earlier rounds.

In round 4 the section was redesigned. Respondents born in 1980-82 (who were mostly age 18 and older when the round 4 field period began) were asked about employee jobs and self-employment at the same time. In addition, the minimum income requirement from the freelance section no longer applied; jobs could be classified as self-employment regardless of earnings. However, respondents born in 1983-84 (who were mostly age 16 or 17 when the round 4 field period began) continued to describe employee and freelance jobs separately. Data on self-employment jobs were still collected in the freelance section, and freelance jobs still had to meet the income criteria to qualify as self-employment.

The redesign of the employment section has important implications for created employment variables. In rounds 1-3, all of the created employment variables were based only on employee-type jobs. So, for example, the variable "Weeks Worked during Calendar Year 1999" counted only the weeks worked by a respondent at a regular employee-type job. If the respondent also reported self-employment in a lawn care business, the weeks spent working at that job were not counted in the created variable.

In round 4, when respondents born in 1980-82 reported both employee-type and self-employed jobs in the same series of variables, this approach was reconsidered. These older respondents now have three versions of most created variables. The first version, identified by the suffix "ET" in the question name, includes only employee-type jobs. The second version, the "SE" variables, includes only self-employed jobs reported by respondents born in 1980-82 in the regular employment section **during round 4**. These variables do **not** include freelance jobs or self-employment reported by younger respondents in the freelance jobs section in round 4, and they do **not** include freelance jobs or self-employment reported in previous rounds by any respondent regardless of age. Finally, the variables for all jobs include both employee-type jobs and self-employment reported during round 4 for respondents born in 1980-82 but only include employee-type jobs for respondents born in 1983-84. These last variables are idenitified with the suffix "ALL" in the question name.

Respondents' ages varied widely in round 4, when self-employed jobs were first recorded as part of the regular employment section, and some respondents reported employment over several years if they missed a round of interviewing. To simplify the creation of the employment variables, survey staff included only the self-employment job information starting on January 1 of the year the respondent turned 18. For example, consider a respondent who was 20 years old on his round 4 interview date in April 2001 and had not been interviewed since round 1. He reports self-employment in a computer repair business beginning on his 17th birthday in March 1998 and continuous employment at a fast-food restaurant since his round 1 interview in 1997. The round 4 created employment variables would include information about the employee-type fast food job dating all the way back to 1997. However, the computer repair business would not be considered

until January 1, 1999 (the first day of the year he turned 18). In other words, the variable "Weeks Worked in Calendar Year 1998" would count only the fast food job and the variable "Weeks Worked Any Job in Calendar Year 1999" would count both the fast food job and the repair business. Similarly, the new variable "Weeks R Was Self-Employed Year 1998" would have a value of –4, or valid skip (because the respondent was not yet 18), but the variable "Weeks R Was Self-Employed Year 1999" would report the weeks the respondent worked at the computer repair business in 1999. This approach will permit users to compare the employment variables across respondents in different rounds with confidence that the job types included are the same for all respondents of a given age.

"**Backreporters.**" Occasionally respondents report a job in the current interview that started before the date of their last interview and should have been reported at that time. Appendix 6 in this document contains a more complete description of the implications of these reports for the created employment event history variables. There are also a number of created employment variables detailed in this appendix. Nearly all of these variables use the information provided about employment previous to the date of last interview. The only exception is the set of CV_WKSWK_DLI variables, which reports the weeks worked since the previous interview date. The current round's variables would not include the backreported information in any case; the previous round's variables are **not** re-created to incorporate this new information.

For example, assume that Jane was interviewed in round 3 on April 15, 2000, and in round 4 on April 15, 2001. In the round 4 interview, she reports for the first time a job that started on April 1, 2000. The 2 weeks worked at that job before April 15, 2000, would **not** be reflected in any round's CV_WKSWK_DLI variables. However, those weeks **would** be counted in other variables. For example, CV_WKSWK_YR.00 would count all the weeks worked at any job in 2000, regardless of whether those weeks were reported in the round 3 or round 4 interview.

Creation of Employment Variables. A number of the created employment variables use the same program as input. The program in this section is referred to throughout the employment variables. For example, to create the "Weeks Worked at Employee Job #x during 19xx" variables, survey staff first run the program below titled "emp_begin.sas" and then run the program included in the weeks worked section of this appendix.

EMP_BEGIN.SAS

This program calculates total weeks worked at each job for each respondent. It converts start and stop dates for jobs and within-job gaps to continuous week numbers, subtracts within-job gaps, and finally counts the total weeks worked. The variables listed here are those employed in any program that uses emp_begin.sas as input; they may not be used in this initial program.

Name in Program	Question Name on CD
Round 1	
r1int_d, _m, _y	YINF-900_D, _M, _Y
birthdy, birthmo, birthyr	KEY!BDATE_D, _M, _Y
id	PUBID
Round 2	
r2int_d, _m, _y	YINTDATE~D, ~M, ~Y
Round 3	
r3int_d, _m, _y	YINTDATE~D, ~M, ~Y
Round 4	
r4int_d, _m, _y	YINTDATE~D, ~M, ~Y
e200	YEMP-200A
e303_1-_4	YEMP-303.01-.04
e239011-e239019	YEMP-23901.01-.09
e245011-e245018	YEMP-24501.01-.08

Name in Program	Question Name on CD
BGDY1_1, BGMO1_1, BGYR1_1	YEMP-102200.01.01~D, ~M, ~Y
BGDY2_1, BGMO2_1, BGYR2_1	YEMP-102200.02.01~D, ~M, ~Y
BGDY3_1, BGMO3_1, BGYR3_1	YEMP-102200.03.01~D, ~M, ~Y
BGDY4_1, BGMO4_1, BGYR4_1	YEMP-102200.04.01~D, ~M, ~Y
BGDY5_1, BGMO5_1, BGYR5_1	YEMP-102200.05.01~D, ~M, ~Y
BGDY6_1, BGMO6_1, BGYR6_1	YEMP-102200.06.01~D, ~M, ~Y
BGDY1_2, BGMO1_2, BGYR1_2	YEMP-102400.01.02~D, ~M, ~Y
BGDY1_3, BGMO1_3, BGYR1_3	YEMP-102400.01.03~D, ~M, ~Y
BGDY1_4, BGMO1_4, BGYR1_4	YEMP-102400.01.04~D, ~M, ~Y
BGDY1_5, BGMO1_5, BGYR1_5	YEMP-102400.01.05~D, ~M, ~Y
BGDY1_6, BGMO1_6, BGYR1_6	YEMP-102400.01.06~D, ~M, ~Y
BGDY1_7, BGMO1_7, BGYR1_7	YEMP-102400.01.07~D, ~M, ~Y
BGDY1_8, BGMO1_8, BGYR1_8	YEMP-102400.01.08~D, ~M, ~Y
BGDY1_9, BGMO1_9, BGYR1_9	YEMP-102400.01.09~D, ~M, ~Y
BGDY1_10, BGMO1_10, BGYR1_10	YEMP-102400.01.10~D, ~M, ~Y
BGDY2_2, BGMO2_2, BGYR2_2	YEMP-102400.02.02~D, ~M, ~Y
BGDY2_3, BGMO2_3, BGYR2_3	YEMP-102400.02.03~D, ~M, ~Y

e344021-e344027	YEMP-34402.01-.07
e344031-e344033	YEMP-34403.01-.03
e344281-e344287	YEMP-34428.01-.07
e226101-e226103	YEMP-22610.01.02-.03.02
e226111-e226113	YEMP-22611.01.03-.03.03
e379021-e379029	YEMP-37902.01-.09
e379041-e379047	YEMP-37904.01-.07
e380001-e380009	YEMP-38000.01-.09
e38000F1-e38000F7	YEMP-38000F.01-.07
e381031-e381033	YEMP-38103.01-.03
e381051-e381055	YEMP-38105.01-.05
e599011-e599018	YEMP-59901.01-.08
e880001-e880009	YEMP-88000.01-.09
e885011-e885017	YEMP-88501.01-.07
e984021-e984027	YEMP-98402.01-.07
e984031-e984033	YEMP-98403.01-.03
e984291-e984293	YEMP-98429.01-.03
e1002311-e1002314	YEMP-100231.01.02-.04.02
e1002321-e1002323	YEMP-100232.01.03-.03.03
GAP1-GAP9	YEMP-101500.01-.09
NUMGAP1-NUMGAP6	YEMP-103200.01-.06
milflag1-milflag9	YEMP_MILFLAG.01-.09
self1-self9	YEMP_SELFEMP.01-.09
stardy1-stardy9	YEMP_STARTDATE.01~D-.09~D
starmo1-starmo9	YEMP_STARTDATE.01~M-.09~M
staryr1-staryr9	YEMP_STARTDATE.01~Y-.09~Y
stopdy1-stopdy9	YEMP_STOPDATE.01~D-.09~D
stopmo1-stopmo9	YEMP_STOPDATE.01~M-.09~M
stopyr1-stopyr9	YEMP_STOPDATE.01~Y-.09~Y
uid1-uid9	YEMP_UID.01-.09
BGDY2_4, BGMO2_4, BGYR2_4	YEMP-102400.02.04~D, ~M, ~Y
BGDY2_5, BGMO2_5, BGYR2_5	YEMP-102400.02.05~D, ~M, ~Y
BGDY3_2, BGMO3_2, BGYR3_2	YEMP-102400.03.02~D, ~M, ~Y
BGDY3_3, BGMO3_3, BGYR3_3	YEMP-102400.03.03~D, ~M, ~Y
BGDY4_2, BGMO4_2, BGYR4_2	YEMP-102400.04.02~D, ~M, ~Y
BGDY4_3, BGMO4_3, BGYR4_3	YEMP-102400.04.03~D, ~M, ~Y
EGDY1_1, EGMO1_1, EGYR1_1	YEMP-102700.01.01~D, ~M, ~Y
EGDY1_2, EGMO1_2, EGYR1_2	YEMP-102700.01.02~D, ~M, ~Y
EGDY1_3, EGMO1_3, EGYR1_3	YEMP-102700.01.03~D, ~M, ~Y
EGDY1_4, EGMO1_4, EGYR1_4	YEMP-102700.01.04~D, ~M, ~Y
EGDY1_5, EGMO1_5, EGYR1_5	YEMP-102700.01.05~D, ~M, ~Y
EGDY1_6, EGMO1_6, EGYR1_6	YEMP-102700.01.06~D, ~M, ~Y
EGDY1_7, EGMO1_7, EGYR1_7	YEMP-102700.01.07~D, ~M, ~Y
EGDY1_8, EGMO1_8, EGYR1_8	YEMP-102700.01.08~D, ~M, ~Y
EGDY1_9, EGMO1_9, EGYR1_9	YEMP-102700.01.09~D, ~M, ~Y
EGDY1_10, EGMO1_10, EGYR1_10	YEMP-102700.01.10~D, ~M, ~Y
EGDY2_1, EGMO2_1, EGYR2_1	YEMP-102700.02.01~D, ~M, ~Y
EGDY2_2, EGMO2_2, EGYR2_2	YEMP-102700.02.02~D, ~M, ~Y
EGDY2_3, EGMO2_3, EGYR2_3	YEMP-102700.02.03~D, ~M, ~Y
EGDY2_4, EGMO2_4, EGYR2_4	YEMP-102700.02.04~D, ~M, ~Y
EGDY2_5, EGMO2_5, EGYR2_5	YEMP-102700.02.05~D, ~M, ~Y
EGDY3_1, EGMO3_1, EGYR3_1	YEMP-102700.03.01~D, ~M, ~Y
EGDY3_2, EGMO3_2, EGYR3_2	YEMP-102700.03.02~D, ~M, ~Y
EGDY3_3, EGMO3_3, EGYR3_3	YEMP-102700.03.03~D, ~M, ~Y
EGDY4_1, EGMO4_1, EGYR4_1	YEMP-102700.04.01~D, ~M, ~Y
EGDY4_2, EGMO4_2, EGYR4_2	YEMP-102700.04.02~D, ~M, ~Y
EGDY4_3, EGMO4_3, EGYR4_3	YEMP-102700.04.03~D, ~M, ~Y
EGDY5_1, EGMO5_1, EGYR5_1	YEMP-102700.05.01~D, ~M, ~Y
EGDY6_1, EGMO6_1, EGYR6_1	YEMP-102700.06.01~D, ~M, ~Y

/************** SECTION 1: Convert birth week, week turned age 14, week turned age 20, ***************\

```
/* Default age 14 week = 9999 */
age14wk=9999;
  if age14yr>0 and bbdays>0 then do;
    if age14yr=1980 then do; age14wk=ceil((bbdays+2)/7); end;
    if age14yr=1981 then do; age14wk=52+ceil((bbdays+4)/7); end;
    if age14yr=1982 then do; age14wk=104+ceil((bbdays+5)/7); end;
    if age14yr=1983 then do; age14wk=156+ceil((bbdays+6)/7); end;
    if age14yr=1984 then do; age14wk=209+ceil((bbdays)/7); end;
    if age14yr=1985 then do; age14wk=261+ceil((bbdays+2)/7); end;
    if age14yr=1986 then do; age14wk=313+ceil((bbdays+3)/7); end;
    if age14yr=1987 then do; age14wk=365+ceil((bbdays+4)/7); end;
    if age14yr=1988 then do; age14wk=417+ceil((bbdays+5)/7); end;
    if age14yr=1989 then do; age14wk=470+ceil((bbdays)/7); end;
    if age14yr=1990 then do; age14wk=522+ceil((bbdays+1)/7); end;
    if age14yr=1991 then do; age14wk=574+ceil((bbdays+2)/7); end;
    if age14yr=1992 then do; age14wk=626+ceil((bbdays+3)/7); end;
    if age14yr=1993 then do; age14wk=678+ceil((bbdays+5)/7); end;
    if age14yr=1994 then do; age14wk=730+ceil((bbdays+6)/7); end;
    if age14yr=1995 then do; age14wk=783+ceil((bbdays)/7); end;
    if age14yr=1996 then do; age14wk=835+ceil((bbdays+1)/7); end;
    if age14yr=1997 then do; age14wk=887+ceil((bbdays+3)/7); end;
    if age14yr=1998 then do; age14wk=939+ceil((bbdays+4)/7); end;
    if age14yr=1999 then do; age14wk=991+ceil((bbdays+5)/7); end;
    if age14yr=2000 then do; age14wk=1043+ceil((bbdays+6)/7); end;
  end;

/** Calulate Age 18 year  **/ AGE18YR=birthyr+18;
** Convert Age 18 Birthdate to week number  ** ***Convert age18 month and day to total days (bbdays);
  if birthmo>0 and birthdy>0 then do;
    if birthmo=1 then bbdays=birthdy;                    if birthmo=2 then bbdays=birthdy+31;
    if birthmo=3 then bbdays=birthdy+59;                 if birthmo=4 then bbdays=birthdy+90;
    if birthmo=5 then bbdays=birthdy+120;                if birthmo=6 then bbdays=birthdy+151;
    if birthmo=7 then bbdays=birthdy+181;                if birthmo=8 then bbdays=birthdy+212;
    if birthmo=9 then bbdays=birthdy+243;                if birthmo=10 then bbdays=birthdy+273;
    if birthmo=11 then bbdays=birthdy+304;               if birthmo=12 then bbdays=birthdy+334;
  end;

***Account for leap years;
  if age18yr=1980 or age18yr=1984 or age18yr=1988 or age18yr=1992 or age18yr=1996 or age18yr=2000 then do;
    if birthmo>0 and birthdy>0 then do;
      if birthmo=1 then bbdays=birthdy;                    if birthmo=2 then bbdays=birthdy+31;
      if birthmo=3 then bbdays=birthdy+60;                 if birthmo=4 then bbdays=birthdy+91;
      if birthmo=5 then bbdays=birthdy+121;                if birthmo=6 then bbdays=birthdy+152;
      if birthmo=7 then bbdays=birthdy+182;                if birthmo=8 then bbdays=birthdy+213;
      if birthmo=9 then bbdays=birthdy+244;                if birthmo=10 then bbdays=birthdy+274;
      if birthmo=11 then bbdays=birthdy+305;               if birthmo=12 then bbdays=birthdy+335;
    end;
  end;

***Convert days into week numbers;  /* Default age 14 week = 9999 */
age18wk=9999;
  if age18yr>0 and bbdays>0 then do;
    if age18yr=1980 then do; age18wk=ceil((bbdays+2)/7); ag18jan=0; end;
    if age18yr=1981 then do; age18wk=52+ceil((bbdays+4)/7); ag18jan=52; end;
    if age18yr=1982 then do; age18wk=104+ceil((bbdays+5)/7); ag18jan=104; end;
    if age18yr=1983 then do; age18wk=156+ceil((bbdays+6)/7); ag18jan=156; end;
    if age18yr=1984 then do; age18wk=209+ceil((bbdays)/7); ag18jan=209; end;
```

```
    if age18yr=1985 then do; age18wk=261+ceil((bbdays+2)/7); ag18jan=261; end;
    if age18yr=1986 then do; age18wk=313+ceil((bbdays+3)/7); ag18jan=313; end;
    if age18yr=1987 then do; age18wk=365+ceil((bbdays+4)/7); ag18jan=365; end;
    if age18yr=1988 then do; age18wk=417+ceil((bbdays+5)/7); ag18jan=417; end;
    if age18yr=1989 then do; age18wk=470+ceil((bbdays)/7); ag18jan=470; end;
    if age18yr=1990 then do; age18wk=522+ceil((bbdays+1)/7); ag18jan=522; end;
    if age18yr=1991 then do; age18wk=574+ceil((bbdays+2)/7); ag18jan=574; end;
    if age18yr=1992 then do; age18wk=626+ceil((bbdays+3)/7); ag18jan=626; end;
    if age18yr=1993 then do; age18wk=678+ceil((bbdays+5)/7); ag18jan=678; end;
    if age18yr=1994 then do; age18wk=730+ceil((bbdays+6)/7); ag18jan=730; end;
    if age18yr=1995 then do; age18wk=783+ceil((bbdays)/7); ag18jan=784; end;
    if age18yr=1996 then do; age18wk=835+ceil((bbdays+1)/7); ag18jan=836; end;
    if age18yr=1997 then do; age18wk=887+ceil((bbdays+3)/7); ag18jan=888; end;
    if age18yr=1998 then do; age18wk=939+ceil((bbdays+4)/7); ag18jan=940; end;
    if age18yr=1999 then do; age18wk=991+ceil((bbdays+5)/7); ag18jan=992; end;
    if age18yr=2000 then do; age18wk=1043+ceil((bbdays+6)/7); ag18jan=1044; end;
    if age18yr=2001 then do; age18wk=1096+ceil((bbdays+1)/7); ag18jan=1097; end;
    if age18yr=2002 then do; age18wk=1148+ceil((bbdays+2)/7); ag18jan=1149; end;
  end;

/** Calulate Age 20 week  **/  AGE20YR=birthyr+20;
** Convert Age 20 Birthdate to week number  ** ***Convert age20 month and day to total days (bbdays);
  if birthmo>0 and birthdy>0 then do;
    if birthmo=1 then bbdays=birthdy;                    if birthmo=2 then bbdays=birthdy+31;
    if birthmo=3 then bbdays=birthdy+59;                 if birthmo=4 then bbdays=birthdy+90;
    if birthmo=5 then bbdays=birthdy+120;                if birthmo=6 then bbdays=birthdy+151;
    if birthmo=7 then bbdays=birthdy+181;                if birthmo=8 then bbdays=birthdy+212;
    if birthmo=9 then bbdays=birthdy+243;                if birthmo=10 then bbdays=birthdy+273;
    if birthmo=11 then bbdays=birthdy+304;               if birthmo=12 then bbdays=birthdy+334;
  end;

***Account for leap years;
  if age18yr=1980 or age18yr=1984 or age18yr=1988 or age18yr=1992 or age18yr=1996 or age18yr=2000 then do;
    if birthmo>0 and birthdy>0 then do;
      if birthmo=1 then bbdays=birthdy;                  if birthmo=2 then bbdays=birthdy+31;
      if birthmo=3 then bbdays=birthdy+60;               if birthmo=4 then bbdays=birthdy+91;
      if birthmo=5 then bbdays=birthdy+121;              if birthmo=6 then bbdays=birthdy+152;
      if birthmo=7 then bbdays=birthdy+182;              if birthmo=8 then bbdays=birthdy+213;
      if birthmo=9 then bbdays=birthdy+244;              if birthmo=10 then bbdays=birthdy+274;
      if birthmo=11 then bbdays=birthdy+305;             if birthmo=12 then bbdays=birthdy+335;
    end;
  end;

***Convert days into week numbers; /* Default age 20 week = 9999 */
age20wk=9999;
  if age20yr>0 and bbdays>0 then do;
    if age20yr=1980 then do; age20wk=ceil((bbdays+2)/7); end;
    if age20yr=1981 then do; age20wk=52+ceil((bbdays+4)/7); end;
    if age20yr=1982 then do; age20wk=104+ceil((bbdays+5)/7); end;
    if age20yr=1983 then do; age20wk=156+ceil((bbdays+6)/7); end;
    if age20yr=1984 then do; age20wk=209+ceil((bbdays)/7); end;
    if age20yr=1985 then do; age20wk=261+ceil((bbdays+2)/7); end;
    if age20yr=1986 then do; age20wk=313+ceil((bbdays+3)/7); end;
    if age20yr=1987 then do; age20wk=365+ceil((bbdays+4)/7); end;
    if age20yr=1988 then do; age20wk=417+ceil((bbdays+5)/7); end;
    if age20yr=1989 then do; age20wk=470+ceil((bbdays)/7); end;
    if age20yr=1990 then do; age20wk=522+ceil((bbdays+1)/7); end;
```

```
    if age20yr=1991 then do; age20wk=574+ceil((bbdays+2)/7); end;
    if age20yr=1992 then do; age20wk=626+ceil((bbdays+3)/7); end;
    if age20yr=1993 then do; age20wk=678+ceil((bbdays+5)/7); end;
    if age20yr=1994 then do; age20wk=730+ceil((bbdays+6)/7); end;
    if age20yr=1995 then do; age20wk=783+ceil((bbdays)/7); end;
    if age20yr=1996 then do; age20wk=835+ceil((bbdays+1)/7); end;
    if age20yr=1997 then do; age20wk=887+ceil((bbdays+3)/7); end;
    if age20yr=1998 then do; age20wk=939+ceil((bbdays+4)/7); end;
    if age20yr=1999 then do; age20wk=991+ceil((bbdays+5)/7); end;
    if age20yr=2000 then do; age20wk=1043+ceil((bbdays+6)/7); end;
    if age20yr=2001 then do; age20wk=1096+ceil((bbdays+1)/7); end;
    if age20yr=2002 then do; age20wk=1148+ceil((bbdays+2)/7); end;
    if age20yr=2003 then do; age20wk=1200+ceil((bbdays+2)/7); end;
    if age20yr=2004 then do; age20wk=1252+ceil((bbdays+2)/7); end;
  end;

/** Convert Birthdate to week number  **/  /* Default birthdate week=0 if birthdate < 12/30/79 */
birthwk=0;
  if birthyr>0 and bbdays>0 then do;
    if birthyr=1980 then do; birthwk=ceil((bbdays+2)/7); end;
    if birthyr=1981 then do; birthwk=52+ceil((bbdays+4)/7); end;
    if birthyr=1982 then do; birthwk=104+ceil((bbdays+5)/7); end;
    if birthyr=1983 then do; birthwk=156+ceil((bbdays+6)/7); end;
    if birthyr=1984 then do; birthwk=209+ceil((bbdays)/7); end;
    if birthyr=1985 then do; birthwk=261+ceil((bbdays+2)/7); end;
    if birthyr=1986 then do; birthwk=313+ceil((bbdays+3)/7); end;
    if birthyr=1987 then do; birthwk=365+ceil((bbdays+4)/7); end;
    if birthyr=1988 then do; birthwk=417+ceil((bbdays+5)/7); end;
    if birthyr=1989 then do; birthwk=470+ceil((bbdays)/7); end;
    if birthyr=1990 then do; birthwk=522+ceil((bbdays+1)/7); end;
    if birthyr=1991 then do; birthwk=574+ceil((bbdays+2)/7); end;
    if birthyr=1992 then do; birthwk=626+ceil((bbdays+3)/7); end;
    if birthyr=1993 then do; birthwk=678+ceil((bbdays+5)/7); end;
    if birthyr=1994 then do; birthwk=730+ceil((bbdays+6)/7); end;
    if birthyr=1995 then do; birthwk=783+ceil((bbdays)/7); end;
    if birthyr=1996 then do; birthwk=835+ceil((bbdays+1)/7); end;
    if birthyr=1997 then do; birthwk=887+ceil((bbdays+3)/7); end;
    if birthyr=1998 then do; birthwk=939+ceil((bbdays+4)/7); end;
    if birthyr=1999 then do; birthwk=991+ceil((bbdays+5)/7); end;
    if birthyr=2000 then do; birthyr=1043+ceil((bbdays+6)/7); end;
  end;

/********** Section 2:  Convert Interview Dates to week number  **********/
***Convert Round 4 interview month and day to total days (intdays);
  if r4int_m>0 and r4int_d>0 then do;
    if r4int_m=1 then intdays=r4int_d;                  if r4int_m=2 then intdays=r4int_d+31;
    if r4int_m=3 then intdays=r4int_d+59;               if r4int_m=4 then intdays=r4int_d+90;
    if r4int_m=5 then intdays=r4int_d+120;              if r4int_m=6 then intdays=r4int_d+151;
    if r4int_m=7 then intdays=r4int_d+181;              if r4int_m=8 then intdays=r4int_d+212;
    if r4int_m=9 then intdays=r4int_d+243;              if r4int_m=10 then intdays=r4int_d+273;
    if r4int_m=11 then intdays=r4int_d+304;             if r4int_m=12 then intdays=r4int_d+334;
  end;

***Account for leap year 2000;
  if r4int_m>0 and r4int_d>0 and r4int_y=2000 then do;
    if r4int_m=1 then intdays=r4int_d;                  if r4int_m=2 then intdays=r4int_d+31;
    if r4int_m=3 then intdays=r4int_d+60;               if r4int_m=4 then intdays=r4int_d+91;
```

```
    if r4int_m=5 then intdays=r4int_d+121;          if r4int_m=6 then intdays=r4int_d+152;
    if r4int_m=7 then intdays=r4int_d+182;          if r4int_m=8 then intdays=r4int_d+213;
    if r4int_m=9 then intdays=r4int_d+244;          if r4int_m=10 then intdays=r4int_d+274;
    if r4int_m=11 then intdays=r4int_d+305;         if r4int_m=12 then intdays=r4int_d+335;
  end;

***Convert days into week numbers; /* Default interview week = 9999 */
intwk=9999;
  if r4int_y>0 and intdays>0 then do;
    if r4int_y=2000 then do; intwk=1043+ceil((intdays+6)/7); end;
    if r4int_y=2001 then do; intwk=1096+ceil((intdays+1)/7); end;
  end;

** Convert Round 3 Interview Date to week number  **; ***Convert interview month and day to total days (intdays);
  if r3int_m>0 and r3int_d>0 then do;
    if r3int_m=1 then intdays=r3int_d;              if r3int_m=2 then intdays=r3int_d+31;
    if r3int_m=3 then intdays=r3int_d+59;           if r3int_m=4 then intdays=r3int_d+90;
    if r3int_m=5 then intdays=r3int_d+120;          if r3int_m=6 then intdays=r3int_d+151;
    if r3int_m=7 then intdays=r3int_d+181;          if r3int_m=8 then intdays=r3int_d+212;
    if r3int_m=9 then intdays=r3int_d+243;          if r3int_m=10 then intdays=r3int_d+273;
    if r3int_m=11 then intdays=r3int_d+304;         if r3int_m=12 then intdays=r3int_d+334;
  end;

***Account for leap year 2000;
  if r3int_m>0 and r3int_d>0 and r3int_y=2000 then do;
    if r3int_m=1 then intdays=r3int_d;              if r3int_m=2 then intdays=r3int_d+31;
    if r3int_m=3 then intdays=r3int_d+60;           if r3int_m=4 then intdays=r3int_d+91;
    if r3int_m=5 then intdays=r3int_d+121;          if r3int_m=6 then intdays=r3int_d+152;
    if r3int_m=7 then intdays=r3int_d+182;          if r3int_m=8 then intdays=r3int_d+213;
    if r3int_m=9 then intdays=r3int_d+244;          if r3int_m=10 then intdays=r3int_d+274;
    if r3int_m=11 then intdays=r3int_d+305;         if r3int_m=12 then intdays=r3int_d+335;
  end;


***Convert days into week numbers;  /* Default interview week = 9999 */
r3int=9999;
  if r3int_y>0 and intdays>0 then do;
    if r3int_y=1999 then do; r3int=991+ceil((intdays+5)/7); end;
    if r3int_y=2000 then do; r3int=1043+ceil((intdays+6)/7); end;
  end;

** Convert Round 2 Date of Interview to week number  ** **Convert interview month and day to total days (rd2days);
  if r2int_m>0 and r2int_d>0 then do;
    if r2int_m=1 then rd2days=r2int_d;              if r2int_m=2 then rd2days=r2int_d+31;
    if r2int_m=3 then rd2days=r2int_d+59;           if r2int_m=4 then rd2days=r2int_d+90;
    if r2int_m=5 then rd2days=r2int_d+120;          if r2int_m=6 then rd2days=r2int_d+151;
    if r2int_m=7 then rd2days=r2int_d+181;          if r2int_m=8 then rd2days=r2int_d+212;
    if r2int_m=9 then rd2days=r2int_d+243;          if r2int_m=10 then rd2days=r2int_d+273;
    if r2int_m=11 then rd2days=r2int_d+304;         if r2int_m=12 then rd2days=r2int_d+334;
  end;

***Convert days into week numbers;  /* Default interview week = 9999 */
r2int=9999;
  if r2int_y>0 and rd2days>0 then do;
    if r2int_y=1998 then do; r2int=939+ceil((rd2days+4)/7); end;
    if r2int_y=1999 then do; r2int=991+ceil((rd2days+5)/7); end;
  end;
```

```
** Convert Date of Round 1 Interview to week number **Convert interview month and day to total days (rd2days);
  if r1int_m>0 and r1int_d>0 then do;
    if r1int_m=1 then rd1days=r1int_d;                    if r1int_m=2 then rd1days=r1int_d+31;
    if r1int_m=3 then rd1days=r1int_d+59;                 if r1int_m=4 then rd1days=r1int_d+90;
    if r1int_m=5 then rd1days=r1int_d+120;                if r1int_m=6 then rd1days=r1int_d+151;
    if r1int_m=7 then rd1days=r1int_d+181;                if r1int_m=8 then rd1days=r1int_d+212;
    if r1int_m=9 then rd1days=r1int_d+243;                if r1int_m=10 then rd1days=r1int_d+273;
    if r1int_m=11 then rd1days=r1int_d+304;               if r1int_m=12 then rd1days=r1int_d+334;
  end;

***Convert days into week numbers; /* Default interview week = 9999 */
r1int=9999;
  if r1int_y>0 and rd1days>0 then do;
    if r1int_y=1997 then do; r1int=887+ceil((rd1days+3)/7); end;
    if r1int_y=1998 then do; r1int=939+ceil((rd1days+4)/7); end;
  end;

/* Hand edits carried over from Round 2 */
if (id=1000291 or id=1562991 or id=1590571) then do; r1int_y=1997; end;
if id in (1057531,1058331) then do; r2int=990; r2int_y=1998; end;
if id=1745011 then do; r1int=919; r1int_y=1997; end;

/* Done to create a seamless flow between rounds */
intwk=intwk-1;

/* Round 2 non-interview case */     if r2int_d=-5 then r2int=-5;
/* Round 3 non-interview case */     if r3int_d=-5 then r3int=-5;
/* Round 4 non-interview case */     if e200=-5 then intwk=-5;

/**************** Section 3: Convert Start/Stop dates into NLSY97 week numbers ****************/
/* This section reads in raw start and stop dates for each job (max=9) and converts them into NLSY97
week numbers. Some start/stop DAYS and MONTHS have been imputed if missing. */
/*ostartm represents the "old" start dates, used when start dates are updated to interview dates. */
array ostartm (i) ostarmo1-ostarmo9;          array ostartd (i) ostardy1-ostardy9;
array ostarty (i) ostaryr1-ostaryr9;          array ostopm (i) ostopmo1-ostopmo9;
array ostopd (i) ostopdy1-ostopdy9;           array ostopy (i) ostopyr1-ostopyr9;
array startm (i) starmo1-starmo9;             array startd (i) stardy1-stardy9;
array starty (i) staryr1-staryr9;             array stopm  (i) stopmo1-stopmo9;
array stopd  (i) stopdy1-stopdy9;             array stopy  (i) stopyr1-stopyr9;
array UID    (i) UID1-UID9;
array sttdays (i) sttday1-sttday9;   /* total days in that year from startdate (to Jan 1) */
array stpdays (i) stpday1-stpday9;   /* total days in that year from stopdate (to Jan 1)  */
array startwk (i) starw1-starw9;              array stopwk  (i) stopw1-stopw9;
array srflag  (i) srflg1-srflg10;             array spflag  (i) spflg1-spflg10;
array uflag   (i) uflag1-uflag9;     /* uflag=1 when job startdate is updated */
array smofl   (i) smofl1-smofl9;     /* dummy equals 1 when a start month is imputed */
array emofl   (i) emofl1-emofl9;     /* dummy equals 1 when a stop month is imputed */
array self   (i) self1-self9;

/* Establish dli interview round */
if r3int_y>0 then do; dli_y=r3int_y; dli_m=r3int_m; dli_d=r3int_d; dliwk=r3int; r3dli=1; end;
if r3int_y=-5 and r2int_y>0 then do; dli_y=r2int_y; dli_m=r2int_m; dli_d=r2int_d; dliwk=r2int; r2dli=1; end;
if r3int_y=-5 and r2int_y=-5 and r1int_y>0 then do; dli_y=r1int_y; dli_m=r1int_m; dli_d=r1int_d; dliwk=r1int;
          r1dli=1; end;

/* Initialize smofl and emofl */          do i=1 to 9; smofl=0; emofl=0; end;
```

```
/* Define old start and stop dates */
do i=1 to 9; ostartm=startm; ostartd=startd; ostarty=starty; ostopm=stopm; ostopd=stopd; ostopy=stopy; end;

***Fill-in start/stop day for those missing;
/*  flag1 = impute start day (valid month)          flag2 = impute start month (valid day)
    flag3 = impute start day and month              flag4 = impute stop day (valid month)
    flag5 = impute stop month (valid day)           flag6 = impute stop day and month        */
do i=1 to 9; flag1=-4; flag2=-4; flag3=-4; flag4=-4; flag5=-4; flag6=-4; end;

/* Impute missing start days to 1, missing stop days to 28, missing start months to 1 (Jan.) and missing stop months to
          12 (Dec.). */ /* Reset flag(1-3) to zero when the start year is valid */
do over starty;
  if starty>0 then do;
    flag1=0; flag2=0; flag3=0;
    if startm>0 and startd<=0 then do; startd=1; flag1=1; srflag=1; end;
    if startm<=0 and startd>0 then do; startm=1; flag2=1; srflag=1; smofl=1; end;
    if startm<=0 and startd<=0 then do; startm=1; startd=1; flag3=1; srflag=1; smofl=1; end;
  end;
end;

do over stopy;
  if stopy>0 then do;
  flag4=0; flag5=0; flag6=0;
    if stopm>0 and stopd<=0 then do; stopd=28; flag4=1; spflag=1; end;
    if stopm<=0 and stopd>0 then do; stopm=12; flag5=1; spflag=1; emofl=1; end;
    if stopm<=0 and stopd<=0 then do; stopm=12; stopd=28; flag6=1; spflag=1; emofl=1; end;
/* The following lines prevent imputed values for stop months and days to exceed the Round4 interview date. */
    if stopy=r4int_y and stopm=r4int_m and stopd>r4int_d and spflag=1 then do; stopd=r4int_d; end;
    if stopy=r4int_y and stopm>r4int_m then do; stopm=r4int_m; stopd=r4int_d; end;
  end;
end;

/* If a respondent backreports a job that begins before the dli but ends after, then the start date is updated to the dli. If a
          respondent backerports a job that begins and ends before the dli, that job is eliminated. A list of the deleted
          jobs is below. Deletion occurs after the start/stop dates are turned into week numbers later in the program.
          The id's of the backreporters are listed below. */
do i=1 to 9;
if 0<starty<dli_y and 0<stopy<dli_y then brep=1;
if starty<dli_y and stopy=dli_y and 0<stopm<dli_m then brep=1;
if starty=dli_y and stopy=dli_y and 0<startm<dli_m and 0<stopm<dli_m then brep=1;
if starty=dli_y and stopy=dli_y and 0<startm<dli_m and stopm=dli_m and 0<stopd<dli_d then brep=1;
if starty=dli_y and stopy=dli_y and startm=dli_m and stopm=dli_m and 0<startd<dli_d and 0<stopd<dli_d then
          brep=1;
end;

array backr (i) backr1-backr9;
do i=1 to 9; backr=0; end;
if id=1001101 and uid2=200001 then backr2=1;        if id=1036631 and uid2=200001 then backr2=1;
if id=1094982 and uid3=200001 then backr3=1;        if id=1206521 and uid1=200001 then backr1=1;
if id=1243411 and uid4=200004 then backr4=1;        if id=1243411 and uid5=200005 then backr5=1;
if id=1283021 and uid4=200001 then backr4=1;        if id=1370661 and uid3=200001 then backr3=1;
if id=1438351 and uid2=200002 then backr2=1;        if id=1445111 and uid2=200002 then backr2=1;
if id=1479601 and uid3=200002 then backr3=1;        if id=1549682 and uid4=200003 then backr4=1;
if id=1558701 and uid2=200002 then backr2=1;        if id=1674502 and uid2=200002 then backr2=1;
if id=1674502 and uid3=200001 then backr3=1;        if id=1718641 and uid3=200001 then backr3=1;
if id=1882171 and uid1=200001 then backr1=1;        if id=1927192 and uid1=200001 then backr1=1;
```

```
if id=1955071 and uid2=200001 then backr2=1;

/* We can erase one backreporting job for id 1622011, uid=9802 since we already have data on that job from Round 3.
        This leaves a total of 17 backreporting respondents and 19 backreporting jobs, all of which have 2000 UID's.
        This data will be used in the job-specific c.v.'s, but not the year specific or age 14 c.v.'s. */
/* id 1283021 is a backerporting respondent who has the job erased later in the program because it is a self-employed
        job that begins and ends before Jan.1, of the year of R's 18th birthday. */
/* id 1704581 reports a 1999 job that begins and ends before dli. */
if id=1622011 or id=1704581 then do;
  stardy1=-4; starmo1=-4; staryr1=-4; stopdy1=-4; stopmo1=-4; stopyr1=-4; end;

/* The following lines of code account for the cases where the respondent has a start date for a job reported in an earlier
        round that comes before the Round 3 (dli) interview date. The idea is to only count the weeks employed from
        dli to today. Then, the activity from previous rounds will be added to the Round 4 activity to get the full
        history. To achieve this, all jobs reported in earlier rounds (UID's begin with 97,98,or 1999) will have their
        startdates updated to the Ruond 3 (dli) interview date. The start week and stop week for these jobs will be
        counted the same. */
do i=1 to 9;
  uflag=0;          /* Initialize uflag */
  if uid>-5 and startd>-4 and backr=0 then do;
    if starty=dli_y and startm=dli_m and startd<dli_d and starty>0 then do; startd=dli_d; uflag=1; end;
    if starty=dli_y and startm<dli_m and starty>0 then do; startm=dli_m; startd=dli_d; uflag=1; end;
    if starty<dli_y and starty>0 then do; starty=dli_y; startm=dli_m; startd=dli_d; uflag=1; end;
  end;
end;

***Convert START month and day to total days;
do over startm;
  if startm>0 and startd>0 then do;
    if startm=1 then sttdays=startd;                    if startm=2 then sttdays=startd+31;
    if startm=3 then sttdays=startd+59;                 if startm=4 then sttdays=startd+90;
    if startm=5 then sttdays=startd+120;                if startm=6 then sttdays=startd+151;
    if startm=7 then sttdays=startd+181;                if startm=8 then sttdays=startd+212;
    if startm=9 then sttdays=startd+243;                if startm=10 then sttdays=startd+273;
    if startm=11 then sttdays=startd+304;               if startm=12 then sttdays=startd+334;
  end;
end;

***Account for leap years;
do over starty;
  if starty=1980 or starty=1984 or starty=1988 or starty=1992 or starty=1996 or starty=2000 then do;
    if startm>0 and startd>0 then do;
      if startm=1 then sttdays=startd;                  if startm=2 then sttdays=startd+31;
      if startm=3 then sttdays=startd+60;               if startm=4 then sttdays=startd+91;
      if startm=5 then sttdays=startd+121;              if startm=6 then sttdays=startd+152;
      if startm=7 then sttdays=startd+182;              if startm=8 then sttdays=startd+213;
      if startm=9 then sttdays=startd+244;              if startm=10 then sttdays=startd+274;
      if startm=11 then sttdays=startd+305;             if startm=12 then sttdays=startd+335;
    end;
  end;
end;

***Convert STOP month and day to total days;
do over stopm;
  if stopm>0 and stopd>0 then do;
    if stopm=1 then stpdays=stopd;                      if stopm=2 then stpdays=stopd+31;
    if stopm=3 then stpdays=stopd+59;                   if stopm=4 then stpdays=stopd+90;
```

```
    if stopm=5 then stpdays=stopd+120;                    if stopm=6 then stpdays=stopd+151;
    if stopm=7 then stpdays=stopd+181;                    if stopm=8 then stpdays=stopd+212;
    if stopm=9 then stpdays=stopd+243;                    if stopm=10 then stpdays=stopd+273;
    if stopm=11 then stpdays=stopd+304;                   if stopm=12 then stpdays=stopd+334;
  end;
end;

***Account for leap years;
do over stopy;
  if stopy=1980 or stopy=1984 or stopy=1988 or stopy=1992 or stopy=1996 or stopy=2000 then do;
    if stopm>0 and stopd>0 then do;
      if stopm=1 then stpdays=stopd;                      if stopm=2 then stpdays=stopd+31;
      if stopm=3 then stpdays=stopd+60;                   if stopm=4 then stpdays=stopd+91;
      if stopm=5 then stpdays=stopd+121;                  if stopm=6 then stpdays=stopd+152;
      if stopm=7 then stpdays=stopd+182;                  if stopm=8 then stpdays=stopd+213;
      if stopm=9 then stpdays=stopd+244;                  if stopm=10 then stpdays=stopd+274;
      if stopm=11 then stpdays=stopd+305;                 if stopm=12 then stpdays=stopd+335;
    end;
  end;
end;

***Convert days into week numbers;
/* Create year flag variable */
array byear (i) byear1-byear9;
array eyear (i) eyear1-eyear9;
do over starty;
  if starty>0 and sttdays>0 then do;
    if starty=1980 then do; startwk=ceil((sttdays+2)/7); end;
    if starty=1981 then do; startwk=52+ceil((sttdays+4)/7); end;
    if starty=1982 then do; startwk=104+ceil((sttdays+5)/7); end;
    if starty=1983 then do; startwk=156+ceil((sttdays+6)/7); end;
    if starty=1984 then do; startwk=209+ceil((sttdays)/7); end;
    if starty=1985 then do; startwk=261+ceil((sttdays+2)/7); end;
    if starty=1986 then do; startwk=313+ceil((sttdays+3)/7); end;
    if starty=1987 then do; startwk=365+ceil((sttdays+4)/7); end;
    if starty=1988 then do; startwk=417+ceil((sttdays+5)/7); end;
    if starty=1989 then do; startwk=470+ceil((sttdays)/7); end;
    if starty=1990 then do; startwk=522+ceil((sttdays+1)/7); end;
    if starty=1991 then do; startwk=574+ceil((sttdays+2)/7); end;
    if starty=1992 then do; startwk=626+ceil((sttdays+3)/7); end;
    if starty=1993 then do; startwk=678+ceil((sttdays+5)/7); end;
    if starty=1994 then do; startwk=730+ceil((sttdays+6)/7); end;
    if starty=1995 then do; startwk=783+ceil((sttdays)/7); end;
    if starty=1996 then do; startwk=835+ceil((sttdays+1)/7); end;
    if starty=1997 then do; startwk=887+ceil((sttdays+3)/7); end;
    if starty=1998 then do; startwk=939+ceil((sttdays+4)/7); end;
    if starty=1999 then do; startwk=991+ceil((sttdays+5)/7); end;
    if starty=2000 then do; startwk=1043+ceil((sttdays+6)/7); end;
    if starty=2001 then do; startwk=1096+ceil((sttdays+1)/7); end;
  end;
  if starty<0 and starty>-4 then do; startwk=-3; end;
  if -4<ostarty<0 then byear=1;
end;

do over stopy;
  if stopy>0 and stpdays>0 then do;
    if stopy=1980 then do; stopwk=ceil((stpdays+2)/7); end;
```

```
    if stopy=1981 then do; stopwk=52+ceil((stpdays+4)/7); end;
    if stopy=1982 then do; stopwk=104+ceil((stpdays+5)/7); end;
    if stopy=1983 then do; stopwk=156+ceil((stpdays+6)/7); end;
    if stopy=1984 then do; stopwk=209+ceil((stpdays)/7); end;
    if stopy=1985 then do; stopwk=261+ceil((stpdays+2)/7); end;
    if stopy=1986 then do; stopwk=313+ceil((stpdays+3)/7); end;
    if stopy=1987 then do; stopwk=365+ceil((stpdays+4)/7); end;
    if stopy=1988 then do; stopwk=417+ceil((stpdays+5)/7); end;
    if stopy=1989 then do; stopwk=470+ceil((stpdays)/7); end;
    if stopy=1990 then do; stopwk=522+ceil((stpdays+1)/7); end;
    if stopy=1991 then do; stopwk=574+ceil((stpdays+2)/7); end;
    if stopy=1992 then do; stopwk=626+ceil((stpdays+3)/7); end;
    if stopy=1993 then do; stopwk=678+ceil((stpdays+5)/7); end;
    if stopy=1994 then do; stopwk=730+ceil((stpdays+6)/7); end;
    if stopy=1995 then do; stopwk=783+ceil((stpdays)/7); end;
    if stopy=1996 then do; stopwk=835+ceil((stpdays+1)/7); end;
    if stopy=1997 then do; stopwk=887+ceil((stpdays+3)/7); end;
    if stopy=1998 then do; stopwk=939+ceil((stpdays+4)/7); end;
    if stopy=1999 then do; stopwk=991+ceil((stpdays+5)/7); end;
    if stopy=2000 then do; stopwk=1043+ceil((stpdays+6)/7); end;
    if stopy=2001 then do; stopwk=1096+ceil((stpdays+1)/7); end;
  end;
  if stopy<0 and stopy>-4 then do; stopwk=-3; end;
  if -4<ostarty<0 then eyear=1;
end;

/* Account for backreporters */
do i=1 to 9;
  if backr=1 then do; startwk=.; stopwk=.; uid=-4; end;
end;

/* To create a seamless list of information from Round3 to Round4, the following lines of code are included. By
        decreasing Round4 interview date by one, (which was done in emp_interview.sas) jobs that are worked up to
        the interview date are also decreased by one. This is done so that the R4 interview week will not be counted
        twice for respondents with a job during that time. The same procedure is used in the previous rounds. */
do over stopwk;
  if stopwk>0 and UID>0 then do;
    if stopwk>intwk then do; stopwk=intwk; end;
  end;
end;

/* The following lines considers jobs that begin the same week as the Round4 interview date. Since we are updating the
Round2 interview week by -1, we need to account for jobs that start in the same week or tenures of -1 will result. */
do over startwk;
  if startwk>0 and UID>0 then do;
    if startwk>intwk then do; startwk=intwk; end;
  end;
end;

/* The start date for self-employment jobs will be the week of Jan.1 of the year of the respondent's 18th birthday. This
        method cuts off 3 jobs entirely (both the start and stop week are before Jan. 1 of the 18th birthday), and
        shortens 19 other jobs. 20 respondents are affected. */
array bself (i) bself1-bself9;
array eself (i) eself1-eself9;
do over startwk;
  if self=1 and 0<startwk<(ag18jan-1) then do; bself=1; bcheck=1; end;
  if self=1 and 0<startwk<(ag18jan-1) and 0<stopwk<(ag18jan-1) then do; eself=1; echeck=1; end;
```

```
end;
do over startwk;
   if bself=1 then do; startwk=ag18jan; end;
   if eself=1 then do; startwk=.; stopwk=.; end;
end;

/* To account for Round 4 non-interview cases */
if starmo1=-5 then do;
   do over startwk; startwk=-5; stopwk=-5; end;
end;

/* Tests for missing start/stop data */
miss=0;
do i=1 to 9;
   if (startm=-4 and stopm>-4) or (startm>-4 and stopm=-4) or (startd=-4 and startd>-4) or (startd>-4 and stopd=-4) or
            (starty=-4 and starty>-4) or (starty>-4 and stopy=-4) then do; miss=1; end;
end;

/* Correcting for imputed values that resulted in the start date being later than the stop date. In these cases, the imputed
            date will be updated to the good date. */
do i=1 to 9;
   if startwk>stopwk and (flag1=1 or flag2=1 or flag3=1) then do; startwk=stopwk; end;
   if startwk>stopwk and (flag4=1 or flag5=1 or flag6=1) then do; stopwk=startwk; end;
end;

/* Check that previously reported jobs do not predate the date of last interview */
do over UID;
   if UID<=200000 and startwk<r1int and (flag1=1 or flag2=1 or flag3=1)  and startwk ne . and r3int=-5 and r2int=-5
            and r1int>0 then do; check=1; end;
   if UID<=200000 and startwk<r2int and (flag1=1 or flag2=1 or flag3=1)  and startwk ne . and r3int=-5 and r2int>0
            then do; check=1; end;
   if UID<=200000 and startwk<r3int and (flag1=1 or flag2=1 or flag3=1)  and startwk ne . and r3int>0 then do;
            check=1; end;
end;

/* Check for imputed start/stop date in incorrect chronological order */
do i=1 to 9;
   if startwk>stopwk and (flag1=1 or flag2=1 or flag3=1) and (flag4=1 or flag5=1 or flag6=1) then do; checky=1; end;
end;

/* Check for back-reporters, i.e. people that report a new job (UID>200000) that begins before the last interview date.*/
do over UID;
   if startwk<r3int and startwk>0 and r3int>0 then do; back3=1; back=1; end;
   if startwk<r2int and startwk>0 and r2int>0 and r3int=-5 then do; back2=1; back=1; end;
   if startwk<r1int and startwk>0 and r1int>0 and r2int=-5 and r3int=-5 then do; back1=1; back=1; end;
end;

/* Check that UID's are different at each loop, when greater than -4. */
do i=2 to 9; if UID1=UID and UID1>-4 then umatch=1; end;
do i=3 to 9; if UID2=UID and UID2>-4 then umatch=1; end;
do i=4 to 9; if UID3=UID and UID3>-4 then umatch=1; end;
do i=5 to 9; if UID4=UID and UID4>-4 then umatch=1; end;
do i=6 to 9; if UID5=UID and UID5>-4 then umatch=1; end;
do i=7 to 9; if UID6=UID and UID6>-4 then umatch=1; end;
do i=8 to 9; if UID7=UID and UID7>-4 then umatch=1; end;
if UID8=UID9 and UID8>-4 then umatch=1;
```

```
/* Check that start/stop dates have positive UID values. */
do i=1 to 9;
   if startwk>0 and UID=-4 then umiss=1;
   if uid>0 and ostarty=-4 then smiss=1;
end;

/* Check to see how many start weeks for self-employed jobs come before the R's 18th birthday. */
array s18ck (i) s18ck1-s18ck9;
array e18ck (i) e18ck1-e18ck9;
do i=1 to 9;
   if startwk>0 and -5<age18wk<9999 and startwk<ag18jan and self=1 and smofl=0 then do; s18ck=1; slook=1; end;
   if stopwk>0 and -5<age18wk<9999 and stopwk<ag18jan and self=1 and emofl=0 then do; e18ck=1; elook=1; end;
end;

array diff (i) diff1-diff9;
do i=1 to 9;
   if s18ck=1 then do; diff=ag18jan-startwk; end;
end;

do i=1 to 9; if (self=1 and 0<uid<199999) then how=1; end;

/* Check for number of people older than 20 at dli (62 in R3, none in previous rounds) and has a job after their 20th
          birthday. (48 at R3).  */
do i=1 to 9;
   if age20wk<r2int then do; why=1; end;
end;

/************** Section 4:  Convert dates of within-job gaps into NLSY97 week numbers. *************/
/**  JOB 1 GAPS  **/
/* Another hand-edit for a data inconsistency. */
if id=1547241 then do; bgdy1_1=-4; bgmo1_1=-4; bgyr1_1=-4; egdy1_1=-4; egmo1_1=-4; egyr1_1=-4; end;

/*  These variables are read as follows:
          BGDY1_1 = Begin day of within-job gap 1 on job 1
          EGMO1_5 = End month of within-job gap 5 on job 1
          BGAP1_3 = Begin week of within-job gap 3 on job 1    [CREATED]     */
array bgdy (i) BGDY1_1-BGDY1_10; array bgmo (i) BGMO1_1-BGMO1_10; array bgyr (i) BGYR1_1-BGYR1_10;
array egdy (i) EGDY1_1-EGDY1_10; array egmo (i) EGMO1_1-EGMO1_10; array egyr (i) EGYR1_1-EGYR1_10;
array bdays (i) bday1_1-bday1_10;      /* begin day of job1_gap#  (internal calculation)*/
array edays (i) eday1_1-eday1_10;      /* end day of job1_gap#    (internal calculation)*/
array bweek (i) bgap1_1-bgap1_10;        /* begin week of job1_gap# (created) */
array eweek (i) egap1_1-egap1_10;        /* end week of job1_gap#   (created) */
array bflag (i) bflg1_1-bflg1_10;
array eflag (i) eflg1_1-eflg1_10;
array bgfl (i) bgfl1_1-bgfl1_10;
array egfl (i) egfl1_1-egfl1_10;

/* only impute start/stop dates if day is missing */  /* Fill-in start day for those missing */
/* Missing gap days (start and stop) are now imputed to 1. Repeated without comment at each job. */
do over bgyr;
   if bgyr>0 then do;
      if bgmo>0 and bgdy<=0 then do; bgdy=1; bflag=1; end;
      /* Account for beginning gap dates before job start date. */
      if bgyr=staryr1 and bgmo=starmo1 and bgdy<stardy1 and bflag=1 then do; bgdy=stardy1; end;
   end;
end;
```

```
/* Fill-in stop day for those missing */
do over egyr;
  if egyr>0 then do;
    if egmo>0 and egdy<=0 then do; egdy=1; eflag=1;
    end;
    /* This accounts for end gap dates after job start date. */
    if egyr=stopyr1 and egmo=stopmo1 and egdy>stopdy1 and eflag=1 then do; egdy=stopdy1; end;
  end;
end;

/*Set flag for gap exists but invalid data*/
do over bgyr;
  bgfl=-4;
  if (-4 < bgmo < 0)  or (-4 < bgyr <0) then do; bgfl=1; end;
end;
do over egyr;
  egfl=-4;
  if (-4 < egyr < 0) or (-4 < egmo < 0) then do; egfl=1; end;
end;

***Identify within-job gaps on JOB 1 ***;
***Convert gap dates to week numbers ***;   ***Convert START month and day to total days (BDAYS);
  do over bgmo;
    if bgmo>0 and bgdy>0 then do;
      if bgmo=1 then bdays=bgdy;                  if bgmo=2 then bdays=bgdy+31;
      if bgmo=3 then bdays=bgdy+59;               if bgmo=4 then bdays=bgdy+90;
      if bgmo=5 then bdays=bgdy+120;              if bgmo=6 then bdays=bgdy+151;
      if bgmo=7 then bdays=bgdy+181;              if bgmo=8 then bdays=bgdy+212;
      if bgmo=9 then bdays=bgdy+243;              if bgmo=10 then bdays=bgdy+273;
      if bgmo=11 then bdays=bgdy+304;             if bgmo=12 then bdays=bgdy+334;
    end;
  end;

  ***Account for leap years;
  do over bgyr;
    if bgyr=1980 or bgyr=1984 or bgyr=1988 or bgyr=1992 or bgyr=1996 or bgyr=2000 then do;
      if bgmo>0 and bgdy>0 then do;
        if bgmo=1 then bdays=bgdy;                  if bgmo=2 then bdays=bgdy+31;
        if bgmo=3 then bdays=bgdy+60;               if bgmo=4 then bdays=bgdy+91;
        if bgmo=5 then bdays=bgdy+121;              if bgmo=6 then bdays=bgdy+152;
        if bgmo=7 then bdays=bgdy+182;              if bgmo=8 then bdays=bgdy+213;
        if bgmo=9 then bdays=bgdy+244;              if bgmo=10 then bdays=bgdy+274;
        if bgmo=11 then bdays=bgdy+305;             if bgmo=12 then bdays=bgdy+335;
      end;
    end;
  end;

  ***Convert STOP month and day to total days (EDAYS);
  do over egmo;
    if egmo>0 and egdy>0 then do;
      if egmo=1 then edays=egdy;                  if egmo=2 then edays=egdy+31;
      if egmo=3 then edays=egdy+59;               if egmo=4 then edays=egdy+90;
      if egmo=5 then edays=egdy+120;              if egmo=6 then edays=egdy+151;
      if egmo=7 then edays=egdy+181;              if egmo=8 then edays=egdy+212;
      if egmo=9 then edays=egdy+243;              if egmo=10 then edays=egdy+273;
      if egmo=11 then edays=egdy+304;             if egmo=12 then edays=egdy+334;
    end;
```

```
end;

***Account for leap years;
do over egyr;
  if egyr=1980 or egyr=1984 or egyr=1988 or egyr=1992 or egyr=1996 or egyr=2000 then do;
    if egmo>0 and egdy>0 then do;
      if egmo=1 then edays=egdy;                if egmo=2 then edays=egdy+31;
      if egmo=3 then edays=egdy+60;             if egmo=4 then edays=egdy+91;
      if egmo=5 then edays=egdy+121;            if egmo=6 then edays=egdy+152;
      if egmo=7 then edays=egdy+182;            if egmo=8 then edays=egdy+213;
      if egmo=9 then edays=egdy+244;            if egmo=10 then edays=egdy+274;
      if egmo=11 then edays=egdy+305;           if egmo=12 then edays=egdy+335;
    end;
  end;
end;

***Convert days into week numbers;
do over bgyr;
  if bgyr>0 and bdays>0 then do;
    if bgyr=1980 then do; bweek=ceil((bdays+2)/7); end;
    if bgyr=1981 then do; bweek=52+ceil((bdays+4)/7); end;
    if bgyr=1982 then do; bweek=104+ceil((bdays+5)/7); end;
    if bgyr=1983 then do; bweek=156+ceil((bdays+6)/7); end;
    if bgyr=1984 then do; bweek=209+ceil((bdays)/7); end;
    if bgyr=1985 then do; bweek=261+ceil((bdays+2)/7); end;
    if bgyr=1986 then do; bweek=313+ceil((bdays+3)/7); end;
    if bgyr=1987 then do; bweek=365+ceil((bdays+4)/7); end;
    if bgyr=1988 then do; bweek=417+ceil((bdays+5)/7); end;
    if bgyr=1989 then do; bweek=470+ceil((bdays)/7); end;
    if bgyr=1990 then do; bweek=522+ceil((bdays+1)/7); end;
    if bgyr=1991 then do; bweek=574+ceil((bdays+2)/7); end;
    if bgyr=1992 then do; bweek=626+ceil((bdays+3)/7); end;
    if bgyr=1993 then do; bweek=678+ceil((bdays+5)/7); end;
    if bgyr=1994 then do; bweek=730+ceil((bdays+6)/7); end;
    if bgyr=1995 then do; bweek=783+ceil((bdays)/7); end;
    if bgyr=1996 then do; bweek=835+ceil((bdays+1)/7); end;
    if bgyr=1997 then do; bweek=887+ceil((bdays+3)/7); end;
    if bgyr=1998 then do; bweek=939+ceil((bdays+4)/7); end;
    if bgyr=1999 then do; bweek=991+ceil((bdays+5)/7); end;
    if bgyr=2000 then do; bweek=1043+ceil((bdays+6)/7); end;
    if bgyr=2001 then do; bweek=1096+ceil((bdays+1)/7); end;
    if bweek>0 then do; bweek=bweek+1; end;
  end;
end;

do over egyr;
  if egyr>0 and edays>0 then do;
    if egyr=1980 then do; eweek=ceil((edays+2)/7); end;
    if egyr=1981 then do; eweek=52+ceil((edays+4)/7); end;
    if egyr=1982 then do; eweek=104+ceil((edays+5)/7); end;
    if egyr=1983 then do; eweek=156+ceil((edays+6)/7); end;
    if egyr=1984 then do; eweek=209+ceil((edays)/7); end;
    if egyr=1985 then do; eweek=261+ceil((edays+2)/7); end;
    if egyr=1986 then do; eweek=313+ceil((edays+3)/7); end;
    if egyr=1987 then do; eweek=365+ceil((edays+4)/7); end;
    if egyr=1988 then do; eweek=417+ceil((edays+5)/7); end;
    if egyr=1989 then do; eweek=470+ceil((edays)/7); end;
```

```
      if egyr=1990 then do; eweek=522+ceil((edays+1)/7); end;
      if egyr=1991 then do; eweek=574+ceil((edays+2)/7); end;
      if egyr=1992 then do; eweek=626+ceil((edays+3)/7); end;
      if egyr=1993 then do; eweek=678+ceil((edays+5)/7); end;
      if egyr=1994 then do; eweek=730+ceil((edays+6)/7); end;
      if egyr=1995 then do; eweek=783+ceil((edays)/7); end;
      if egyr=1996 then do; eweek=835+ceil((edays+1)/7); end;
      if egyr=1997 then do; eweek=887+ceil((edays+3)/7); end;
      if egyr=1998 then do; eweek=939+ceil((edays+4)/7); end;
      if egyr=1999 then do; eweek=991+ceil((edays+5)/7); end;
      if egyr=2000 then do; eweek=1043+ceil((edays+6)/7); end;
      if egyr=2001 then do; eweek=1096+ceil((edays+1)/7); end;
      if eweek>0 then do; eweek=eweek-1; end;
    end;
  end;

array update (i) update1-update10;

/* The following lines omit gap start and stop dates for gaps less than one work week (5 days) */
do over bdays;
  if edays-bdays<5 and bweek>eweek and bdays ne . and edays ne . then do; bweek=.; eweek=.; end;
end;

/* The following lines are for end gap dates that exceed the Round4 interview date due to our rounding methods. */
do over eweek;
  if 0<eweek<1400 and eweek>intwk then do; eweek=intwk; end;
end;

/* The following omits cases where bweek>eweek, which are caused when missing values are substituted in. For
        example, when the day of a beginning gap is unknown, the program uses the 1st. This can cause bweek>eweek,
        which will cause problems when writing programs for the created variables. This situation will be fixed by
        making eweek and bweek the same. This is repeated (without comment) for each job. */
do over bweek;
  if 0<eweek<1200 then do;
    if bweek>eweek then do; bweek=eweek; update=1; end;
  end;
end;

/* If a repsondent reports a gap on previous Round's job that occured before the dli week, then we will not use this
        information.  However, a newly reported job (i.e., 2000 UID) containing within job gaps that occured before
        the dli week will be counted. This is repeated (without comment) for each job. Note that we have respondents
        who were not interviewed in previous round(s). The same process is done for these people, except we use the
        date of last interview. */
/* If a gap begins before the Round 3 interview date and ends after the Round 3 interview date, the beginning gap date
        is updated to the Round 3 interview date. This is repeated at each job without comment. */
do over bweek;
  if UID1>0 and dliwk>0 and bweek>0 and eweek>0 and backr1 ne 1 then do;
/* Entire gap is before dli. */
    if dliwk>bweek and dliwk>eweek then do; bweek=.; eweek=.; end;
/* B.gap is before dli, e.gap is after dli. */
    if dliwk<eweek and dliwk>bweek then do; bweek=dliwk; end;
  end;
end;

/* The following erases gaps that occur after the job has ended, only in cases where there are no imputed job start or
        stop months or years (imputed job start/stop days acceptable) */
do over bweek;
```

```
  if starw1>0 and stopw1>0 and smofl1=0 and emofl1=0 then do;
    if bweek>stopw1 then do; bweek=.; end;
    if eweek>stopw1 then do; eweek=.; end;
  end;
end;

/* To correct for bad gap information.*/
do over bdays;
  if eweek>stopw1 and eweek ne . and stopw1>0 then eweek=stopw1;
  if bweek<starw1 and bweek ne . and bweek>0 then bweek=starw1;
  if bweek>stopw1 and bweek ne . and stopw1>0 then huh=1;
  if eweek<starw1 and eweek ne . and starw1>0 then huh=1;
end;

/*At this point the program loops through the same code for gaps on jobs 2-9.  Due to space considerations, this code is
        not repeated here.  For more information please contact NLS User Services.*/

/* The following hand edits are umatch cases, where two identical UID's are in the Round 4 roster. These are the same
        jobs, and the following condenses them into one job with a within-job gap. */
if id=1216571 then do; starw1=1055; bgap1_1=1079; egap1_1=1086; stopw1=1095; starw2=.; stopw2=.; uid2=-4; end;
if id=1273771 then do; starw1=1075; bgap1_1=1078; egap1_1=1091; stopw1=1112; starw4=.; stopw4=.; uid4=-4; end;
if id=1278971 then do; starw1=1046; bgap1_1=1063; egap1_1=1080; stopw1=1096; bgap1_2=1095; egap1_2=1096;
        starw4=.; stopw4=.; uid4=-4; end;
if id=1329391 then do; starw1=1044; bgap1_1=1047; egap1_1=1067; stopw1=1079; starw3=.; stopw3=.; uid3=-4; end;
if id=1417821 then do; starw3=1041; bgap3_1=1050; egap3_1=1064; stopw3=1072; starw4=.; stopw4=.; uid4=-4; end;

/************************ Section 5:  Weeks worked by respondent on each job ************************/
/* This section counts the weeks worked by the respondent and removes the within job gaps. It will place a "1" into
        weeks where the respondent was employed. Weeks range from the first week of 1980 to the last week of 2001,
        for a total of 1148 weeks. */

array job1wks (i) wk1_1-wk1_1148;          array job2wks (i) wk2_1-wk2_1148;
array job3wks (i) wk3_1-wk3_1148;          array job4wks (i) wk4_1-wk4_1148;
array job5wks (i) wk5_1-wk5_1148;          array job6wks (i) wk6_1-wk6_1148;
array job7wks (i) wk7_1-wk7_1148;          array job8wks (i) wk8_1-wk8_1148;
array job9wks (i) wk9_1-wk9_1148;
array starw (i) starw1-starw9;             array stopw (i) stopw1-stopw9;

* Default Settings;
do i=1 to 1148;
  job1wks=0;     job2wks=0;     job3wks=0;     job4wks=0;     job5wks=0;     job6wks=0;
  job7wks=0;     job8wks=0;     job9wks=0;
end;

* Change military start and stop dates to missing since only civilian jobs are counted;
if milflag1=1 then do; starw1=.; stopw1=.; end;
if milflag2=1 then do; starw2=.; stopw2=.; end;
if milflag3=1 then do; starw3=.; stopw3=.; end;
if milflag4=1 then do; starw4=.; stopw4=.; end;
if milflag5=1 then do; starw5=.; stopw5=.; end;
if milflag6=1 then do; starw6=.; stopw6=.; end;
if milflag7=1 then do; starw7=.; stopw7=.; end;
if milflag8=1 then do; starw8=.; stopw8=.; end;
if milflag9=1 then do; starw9=.; stopw9=.; end;

/* Define rd4wk as the maximum of dliwk and age14wk. This is used for bad start/stop weeks. For the re-interview
        cases, rd3wk will be set to the maximum of the Round 1 interview date and the respondent's 14th birthday. */
```

```
if dliwk>0 then do;
   if dliwk>age14wk then do; rd3wk=dliwk; end;
   if age14wk=>dliwk then do; rd3wk=age14wk; end;
end;

/*** TOTAL WEEKS WORKED ON JOB 1 ***/
starfl_1=0; stopfl_1=0;

if starw1=-3 and uid1 ne -5 then do; starw1=rd3wk; starfl_1=1; end;
if stopw1=-3 and uid1 ne -5 then do; stopw1=intwk; stopfl_1=1; end;
if smofl1=1 then do; starfl_1=1; end;
if emofl1=1 then do; stopfl_1=1; end;

if starw1>0 and stopw1>0 then do;
   do i=(starw1) to (stopw1); job1wks=1; end;
   *** Remove gap 1 on job 1;
   if bgap1_1>0 & egap1_1>0 then do; do i=(bgap1_1) to (egap1_1); job1wks=0; end; end;
   *** Remove gap 2 on job 1;
   if bgap1_2>0 & egap1_2>0 then do; do i=(bgap1_2) to (egap1_2); job1wks=0; end; end;
   *** Remove gap 3 on job 1;
   if bgap1_3>0 & egap1_3>0 then do; do i=(bgap1_3) to (egap1_3); job1wks=0; end; end;
   *** Remove gap 4 on job 1;
   if bgap1_4>0 & egap1_4>0 then do; do i=(bgap1_4) to (egap1_4); job1wks=0; end; end;
   *** Remove gap 5 on job 1;
   if bgap1_5>0 & egap1_5>0 then do; do i=(bgap1_5) to (egap1_5); job1wks=0; end; end;
   *** Remove gap 6 on job 1;
   if bgap1_6>0 & egap1_6>0 then do; do i=(bgap1_6) to (egap1_6); job1wks=0; end; end;
   *** Remove gap 7 on job 1;
   if bgap1_7>0 & egap1_7>0 then do; do i=(bgap1_7) to (egap1_7); job1wks=0; end; end;
   *** Remove gap 8 on job 1;
   if bgap1_8>0 & egap1_8>0 then do; do i=(bgap1_8) to (egap1_8); job1wks=0; end; end;
   *** Remove gap 9 on job 1;
   if bgap1_9>0 & egap1_9>0 then do; do i=(bgap1_9) to (egap1_9); job1wks=0; end; end;
   *** Remove gap 10 on job 1;
   if bgap1_10>0 & egap1_10>0 then do; do i=(bgap1_10) to (egap1_10); job1wks=0; end; end;
   *** Remove gap 1 on job 1 - beginning gap date bad;
   if bgfl1_1=1 & egap1_1>0 then do; do i=(starw1) to (egap1_1); job1wks=-3; gpfl1_1=1; end; end;
   *** Remove gap 1 on job 1 - end gap date bad;
   if bgap1_1>0 & egfl1_1=1 then do; do i=(bgap1_1) to min(stopw1, bgap1_2-1, bgap1_3-1, bgap1_4-1, bgap1_5-1,
         bgap1_6-1, bgap1_7-1, bgap1_8-1, bgap1_9-1, bgap1_10-1, egap1_2+1, egap1_3+1, egap1_4+1, egap1_5+1,
         egap1_6+1, egap1_7+1, egap1_8+1, egap1_9+1, egap1_10+1); job1wks=-3; gpfl1_1=1; end; end;
   *** Remove gap 1 on job 1 - both gap dates bad;
   if bgfl1_1=1 & egfl1_1=1 then do; do i=(starw1) to min(stopw1, bgap1_2-1, bgap1_3-1, bgap1_4-1, bgap1_5-1,
         bgap1_6-1, bgap1_7-1, bgap1_8-1, bgap1_9-1, bgap1_10-1, egap1_2+1, egap1_3+1, egap1_4+1, egap1_5+1,
         egap1_6+1, egap1_7+1, egap1_8+1, egap1_9+1, egap1_10+1); job1wks=-3; gpfl1_1=1; end; end;
   *** Remove gap 2 on job 1 - beginning gap date bad;
   if bgfl1_2=1 & egap1_2>0 then do; do i=max(starw1,bgap1_1-1,egap1_1+1) to (egap1_2); job1wks=-3; gpfl1_2=1;
         end; end;
   *** Remove gap 2 on job 1 - end gap date bad;
   if bgap1_2>0 & egfl1_2=1 then do; do i=(bgap1_2) to min(stopw1, bgap1_3-1, bgap1_4-1, bgap1_5-1, bgap1_6-1,
         bgap1_7-1, bgap1_8-1, bgap1_9-1, bgap1_10-1, egap1_3+1, egap1_4+1, egap1_5+1, egap1_6+1, egap1_7+1,
         egap1_8+1, egap1_9+1, egap1_10+1); job1wks=-3; gpfl1_2=1; end; end;
   *** Remove gap 2 on job 1 - both gap dates bad;
   if bgfl1_2=1 & egfl1_2=1 then do; do i=max(starw1,bgap1_1-1,egap1_1+1) to min(stopw1, bgap1_3-1, bgap1_4-1,
         bgap1_5-1, bgap1_6-1, bgap1_7-1, bgap1_8-1, bgap1_9-1, bgap1_10-1, egap1_3+1, egap1_4+1, egap1_5+1,
         egap1_6+1, egap1_7+1, egap1_8+1, egap1_9+1, egap1_10+1); job1wks=-3; gpfl1_2=1; end; end;
   *** Remove gap 3 on job 1 - beginning gap date bad;
```

```
if bgfl1_3=1 & egap1_3>0 then do; do i=max(starw1,bgap1_1-1,bgap1_2-1,egap1_1+1,egap1_2+1) to (egap1_3);
        job1wks=-3; gpfl1_3=1; end; end;
*** Remove gap 3 on job 1 - end gap date bad;
if bgap1_3>0 & egfl1_3=1 then do; do i=(bgap1_3) to min(stopw1, bgap1_4-1, bgap1_5-1, bgap1_6-1, bgap1_7-1,
        bgap1_8-1, bgap1_9-1, bgap1_10-1, egap1_4+1, egap1_5+1, egap1_6+1, egap1_7+1, egap1_8+1, egap1_9+1,
        egap1_10+1); job1wks=-3; gpfl1_3=1; end; end;
*** Remove gap 3 on job 1 - both gap dates bad;
if bgfl1_3=1 & egfl1_3=1 then do; do i=max(starw1, bgap1_1-1, bgap1_2-1, egap1_1+1, egap1_2+1) to min(stopw1,
        bgap1_4-1, bgap1_5-1, bgap1_6-1, bgap1_7-1, bgap1_8-1, bgap1_9-1, bgap1_10-1, egap1_4+1, egap1_5+1,
        egap1_6+1, egap1_7+1, egap1_8+1, egap1_9+1, egap1_10+1); job1wks=-3; gpfl1_3=1; end; end;
*** Remove gap 4 on job 1 - beginning gap date bad;
if bgfl1_4=1 & egap1_4>0 then do; do i=max(starw1, bgap1_1-1, bgap1_2-1, bgap1_3-1, egap1_1+1, egap1_2+1,
        egap1_3+1) to (egap1_4); job1wks=-3; gpfl1_4=1; end; end;
*** Remove gap 4 on job 1 - end gap date bad;
if bgap1_4>0 & egfl1_4=1 then do; do i=(bgap1_4) to min(stopw1, bgap1_5-1, bgap1_6-1, bgap1_7-1, bgap1_8-1,
        bgap1_9-1, bgap1_10-1, egap1_5+1, egap1_6+1, egap1_7+1, egap1_8+1, egap1_9+1, egap1_10+1);
        job1wks=-3; gpfl1_4=1; end; end;
*** Remove gap 4 on job 1 - both gap dates bad;
if bgfl1_4=1 & egfl1_4=1 then do; do i=max(starw1, bgap1_1-1, bgap1_2-1, bgap1_3-1, egap1_1+1, egap1_2+1,
        egap1_3+1) to min(stopw1, bgap1_5-1, bgap1_6-1, bgap1_7-1, bgap1_8-1, bgap1_9-1, bgap1_10-1,
        egap1_5+1, egap1_6+1, egap1_7+1, egap1_8+1, egap1_9+1, egap1_10+1); job1wks=-3; gpfl1_4=1; end; end;
*** Remove gap 5 on job 1 - beginning gap date bad;
if bgfl1_5=1 & egap1_5>0 then do; do i=max(starw1, bgap1_1-1, bgap1_2-1, bgap1_3-1, bgap1_4-1, egap1_1+1,
        egap1_2+1, egap1_3+1, egap1_4+1) to (egap1_5); job1wks=-3; gpfl1_5=1; end; end;
*** Remove gap 5 on job 1 - end gap date bad;
if bgap1_5>0 & egfl1_5=1 then do; do i=(bgap1_5) to min(stopw1, bgap1_6-1, bgap1_7-1, bgap1_8-1, bgap1_9-1,
        bgap1_10-1, egap1_6+1, egap1_7+1, egap1_8+1, egap1_9+1,egap1_10+1); job1wks=-3; gpfl1_5=1; end; end;
*** Remove gap 5 on job 1 - both gap dates bad;
if bgfl1_5=1 & egfl1_5=1 then do; do i=max(starw1, bgap1_1-1, bgap1_2-1, bgap1_3-1, bgap1_4-1, egap1_1+1,
        egap1_2+1, egap1_3+1, egap1_4+1) to min(stopw1, bgap1_6-1, bgap1_7-1, bgap1_8-1, bgap1_9-1,
        bgap1_10-1, egap1_6+1, egap1_7+1, egap1_8+1, egap1_9+1,egap1_10+1); job1wks=-3; gpfl1_5=1; end; end;
*** Remove gap 6 on job 1 - beginning gap date bad;
if bgfl1_6=1 & egap1_6>0 then do; do i=max(starw1, bgap1_1-1, bgap1_2-1, bgap1_3-1, bgap1_4-1, bgap1_5-1,
        egap1_1+1, egap1_2+1, egap1_3+1, egap1_4+1, egap1_5+1) to (egap1_6); job1wks=-3; gpfl1_6=1; end; end;
*** Remove gap 6 on job 1 - end gap date bad;
if bgap1_6>0 & egfl1_6=1 then do; do i=(bgap1_6) to min(stopw1, bgap1_7-1, bgap1_8-1, bgap1_9-1, bgap1_10-1,
        egap1_7+1, egap1_8+1, egap1_9+1,egap1_10+1); job1wks=-3; gpfl1_6=1; end; end;
*** Remove gap 6 on job 1 - both gap dates bad;
if bgfl1_6=1 & egfl1_6=1 then do; do i=max(starw1, bgap1_1-1, bgap1_2-1, bgap1_3-1, bgap1_4-1, bgap1_5-1,
        egap1_1+1, egap1_2+1, egap1_3+1, egap1_4+1, egap1_5+1) to min(stopw1, bgap1_6-1, bgap1_7-1,
        bgap1_8-1, bgap1_9-1, bgap1_10-1, egap1_6+1, egap1_7+1, egap1_8+1, egap1_9+1, egap1_10+1);
        job1wks=-3; gpfl1_6=1; end; end;
*** Remove gap 7 on job 1 - beginning gap date bad;
if bgfl1_7=1 & egap1_7>0 then do; do i=max(starw1, bgap1_1-1, bgap1_2-1, bgap1_3-1, bgap1_4-1, bgap1_5-1,
        bgap1_6-1, egap1_1+1, egap1_2+1, egap1_3+1, egap1_4+1, egap1_5+1,egap1_6+1) to (egap1_7);
        job1wks=-3; gpfl1_7=1; end; end;
*** Remove gap 7 on job 1 - end gap date bad;
if bgap1_7>0 & egfl1_7=1 then do; do i=(bgap1_7) to min(stopw1, bgap1_8-1, bgap1_9-1, bgap1_10-1, egap1_8+1,
        egap1_9+1, egap1_10+1); job1wks=-3; gpfl1_7=1; end; end;
*** Remove gap 7 on job 1 - both gap dates bad;
if bgfl1_7=1 & egfl1_7=1 then do; do i=max(starw1, bgap1_1-1, bgap1_2-1, bgap1_3-1, bgap1_4-1, bgap1_5-1,
        bgap1_6-1, egap1_1+1, egap1_2+1, egap1_3+1, egap1_4+1, egap1_5+1,egap1_6+1) to min(stopw1,
        bgap1_8-1, bgap1_9-1, bgap1_10-1, egap1_8+1, egap1_9+1, egap1_10+1); job1wks=-3; gpfl1_7=1; end; end;
*** Remove gap 8 on job 1 - beginning gap date bad;
if bgfl1_8=1 & egap1_8>0 then do; do i=max(starw1, bgap1_1-1, bgap1_2-1, bgap1_3-1, bgap1_4-1, bgap1_5-1,
        bgap1_6-1, bgap1_7-1, egap1_1+1, egap1_2+1, egap1_3+1, egap1_4+1, egap1_5+1, egap1_6+1, egap1_7+1)
        to (egap1_8); job1wks=-3; gpfl1_8=1; end; end;
```

```
  *** Remove gap 8 on job 1 - end gap date bad;
  if bgap1_8>0 & egfl1_8=1 then do; do i=(bgap1_8) to min(stopw1, bgap1_9-1, bgap1_10-1, egap1_9+1,
          egap1_10+1); job1wks=-3; gpfl1_8=1; end; end;
  *** Remove gap 8 on job 1 - both gap dates bad;
  if bgfl1_8=1 & egfl1_8=1 then do; do i=max(starw1, bgap1_1-1, bgap1_2-1, bgap1_3-1, bgap1_4-1, bgap1_5-1,
          bgap1_6-1, bgap1_7-1, egap1_1+1, egap1_2+1, egap1_3+1, egap1_4+1, egap1_5+1, egap1_6+1, egap1_7+1)
          to min(stopw1, bgap1_9-1, bgap1_10-1, egap1_9+1, egap1_10+1); job1wks=-3; gpfl1_8=1; end; end;
  *** Remove gap 9 on job 1 - beginning gap date bad;
  if bgfl1_9=1 & egap1_9>0 then do; do i=max(starw1, bgap1_1-1, bgap1_2-1, bgap1_3-1, bgap1_4-1, bgap1_5-1,
          bgap1_6-1, bgap1_7-1, bgap1_8-1, egap1_1+1, egap1_2+1, egap1_3+1, egap1_4+1, egap1_5+1, egap1_6+1,
          egap1_7+1, egap1_8+1) to (egap1_9); job1wks=-3; gpfl1_9=1; end; end;
  *** Remove gap 9 on job 1 - end gap date bad;
  if bgap1_9>0 & egfl1_9=1 then do; do i=(bgap1_9) to min(stopw1,bgap1_10-1,egap1_10+1); job1wks=-3;
          gpfl1_9=1; end; end;
  *** Remove gap 9 on job 1 - both gap dates bad;
  if bgfl1_9=1 & egfl1_9=1 then do; do i=max(starw1, bgap1_1-1, bgap1_2-1, bgap1_3-1, bgap1_4-1, bgap1_5-1,
          bgap1_6-1, bgap1_7-1, bgap1_8-1, egap1_1+1, egap1_2+1, egap1_3+1, egap1_4+1, egap1_5+1, egap1_6+1,
          egap1_7+1, egap1_8+1) to min(stopw1, bgap1_10-1, egap1_10+1); job1wks=-3; gpfl1_9=1; end; end;
  *** Remove gap 10 on job 1 - beginning gap date bad;
  if bgfl1_10=1 & egap1_10>0 then do; do i=max(starw1, bgap1_1-1, bgap1_2-1, bgap1_3-1, bgap1_4-1, bgap1_5-1,
          bgap1_6-1, bgap1_7-1, bgap1_8-1, bgap1_9-1, egap1_1+1, egap1_2+1, egap1_3+1, egap1_4+1, egap1_5+1,
          egap1_6+1, egap1_7+1, egap1_8+1, egap1_9+1) to (egap1_10); job1wks=-3; gpfl1_10=1; end; end;
  *** Remove gap 10 on job 1 - end gap date bad;
  if bgap1_10>0 & egfl1_10=1 then do; do i=(bgap1_10) to (stopw1); job1wks=-3; gpfl1_10=1; end; end;
  *** Remove gap 10 on job 1 - both gap dates bad;
  if bgfl1_10=1 & egfl1_10=1 then do; do i=max(starw1, bgap1_1-1, bgap1_2-1, bgap1_3-1, bgap1_4-1, bgap1_5-1,
          bgap1_6-1, bgap1_7-1, bgap1_8-1, bgap1_9-1, egap1_1+1, egap1_2+1, egap1_3+1, egap1_4+1, egap1_5+1,
          egap1_6+1, egap1_7+1, egap1_8+1, egap1_9+1) to (stopw1); job1wks=-3; gpfl1_10=1; end; end;
end;

if starfl_1=1 then do;
  do i=(starw1) to min(stopw1, bgap1_1-1, bgap1_2-1, bgap1_3-1, bgap1_4-1, bgap1_5-1, bgap1_6-1, bgap1_7-1,
          bgap1_8-1, bgap1_9-1, bgap1_10-1, egap1_1+1, egap1_2+1, egap1_3+1, egap1_4+1, egap1_5+1, egap1_6+1,
          egap1_7+1, egap1_8+1, egap1_9+1, egap1_10+1); job1wks=-3; end;
end;

if stopfl_1=1 then do;
  do i=max(stopw1, bgap1_1-1, bgap1_2-1, bgap1_3-1, bgap1_4-1, bgap1_5-1, bgap1_6-1, bgap1_7-1, bgap1_8-1,
          bgap1_9-1, bgap1_10-1, egap1_1+1, egap1_2+1, egap1_3+1, egap1_4+1, egap1_5+1, egap1_6+1,
          egap1_7+1, egap1_8+1, egap1_9+1, egap1_10+1) to (stopw1); job1wks=-3; end;
end;

/*At this point the program repeats the above code for jobs 2-9.  This code is omitted due to space considerations. For
          more information, please contact NLS User Services.*/

endsas;
```

NUMBER OF WEEKS WORKED DURING 19XX/20XX

Variables Created: CV_WKSWK_YR_ALL.80 – CV_WKSWK_YR_ALL.01
CV_WKSWK_YR_ET.80 – CV_WKSWK_YR_ET.01
CV_WKSWK_YR_SE.98 – CV_WKSWK_YR_SE.01

Programs Used

This program uses **emp_begin.sas** as input (see the first page of this appendix for details).

This program counts the number of weeks each respondent worked at any job for each year of potential work activity (1980-2001). Three sets of variables are created: one for all jobs, one for employee-type jobs, and one for self-employed jobs. **See the introduction to this appendix for important information about the universes for each of these variables.** Respondents not working in a given year are given a default value of zero (0) weeks worked. Otherwise, the variable indicates the actual cumulative weeks worked on all jobs in that year.

```
/* Section 1: Create variables for # of weeks worked at any
job during a given year based on round 3 data. */

array job1wks (i) wk1_1-wk1_1148;
array job2wks (i) wk2_1-wk2_1148;
array job3wks (i) wk3_1-wk3_1148;
array job4wks (i) wk4_1-wk4_1148;
array job5wks (i) wk5_1-wk5_1148;
array job6wks (i) wk6_1-wk6_1148;
array job7wks (i) wk7_1-wk7_1148;
array job8wks (i) wk8_1-wk8_1148;
array job9wks (i) wk9_1-wk9_1148;
array alljobs (i) wks1-wks1148;
array selfjobs (i) swks1-swks1148;
array empjobs (i) ewks1-ewks1148;
array starw (i) starw1-starw9;
array stopw (i) stopw1-stopw9;
array starfl (i) starfl_1-starfl_9;
array stopfl (i) stopfl_1-stopfl_9;
array uid (i) uid1-uid9;
array self (i) self1-self9;
array r4wks (i) r4wks80-r4wks99 r4wks00 r4wks01;
array r4swks (i) r4swks80-r4swks99 r4swks00 r4swks01;
array r4ewks (i) r4ewks80-r4ewks99 r4ewks00 r4ewks01;

/* Create a separate array for self employed jobs and
employee jobs for the new list of created variables due in
Round 4. The arrays are laid down in the same manner as
"alljobs," with the self-employed dummy deciding which
jobs are laid in each array. */
do over alljobs;
  alljobs=0; selfjobs=0; empjobs=0;
end;

/** Overlay multiple jobs over JOB 1 work weeks      **/

do over job1wks;
  alljobs=job1wks;
  if self1=1 then do; selfjobs=job1wks; end;
  if self1=0 then do; empjobs=job1wks; end;
end;

do over alljobs;
  if job2wks=1 then do; alljobs=job2wks; end;
  if job2wks=1 and self2=1 then do; selfjobs=job2wks;
      end;
  if job2wks=1 and self2=0 then do; empjobs=job2wks;
      end;
  if job2wks=-3 and alljobs=0 then do; alljobs=job2wks;
      end;
  if job2wks=-3 and selfjobs=0 and self2=1 then do;
      selfjobs=job2wks; end;
  if job2wks=-3 and empjobs=0 and self2=0 then do;
      empjobs=job2wks; end;
end;

/* At this point the program loops through the job2 code
      for jobs 3-9. This code is omitted here. Contact
      NLS User Services for more information.*/

/**Calculate cumulative weeks on all jobs for each year **/
/* Initialize each annual c.v. */
do over r4wks;
  r4wks=0; r4swks=0; r4ewks=0;
end;

/* Need separate do loops for valid data ("1s") and for
invalid data ("-3s") or it will just add them up */
/* 1980 */
do i=1 to 52;
  if alljobs=1 then do; r4wks80=r4wks80+1; end;
  if selfjobs=1 then do; r4swks80=r4swks80+1; end;
  if empjobs=1 then do; r4ewks80=r4ewks80+1; end;
  if alljobs=-3 then do; r4wks80=-3; end;
  if selfjobs=-3 then do; r4swks80=-3; end;
  if empjobs=-3 then do; r4ewks80=-3; end;
end;

/*********At this point the program loops through the
same code used above for 1980 for each year 1981-2001,
creating the variables r4wks81, r4swks81, and r4ewks81;
r4wks81, r4swks81, and r4ewks81; and so on through
```

r4wks01, r4swks01, and r4ewks01. These loops are deleted due to space considerations; users who need the entire code should contact NLS User Services. The week numbers for the "do i" statement for each year are as follows:

1981	53-104	1992	627-678
1982	105-156	1993	679-730
1983	157-209	1994	731-783
1984	210-261	1995	784-835
1985	262-313	1996	836-887
1986	314-365	1997	888-939
1987	366-417	1998	940-991
1988	418-470	1999	992-1044
1989	471-522	2000	1045-1096
1990	523-574	2001	1097-1148***/
1991	575-626		

```
do i=1 to 9;
/* start date invalid */
if starfl=1 then do;
  if stopw>1 and self=1 and r4wks80>0 then do;
      r4wks80=-3; r4swks80=-3; end;
  if stopw>1 and self=0 and r4wks80>0 then do;
      r4wks80=-3; r4ewks80=-3; end;
  if stopw>52 and self=1 and r4wks81>0 then do;
      r4wks81=-3; r4swks81=-3; end;
  if stopw>52 and self=0 and r4wks81>0 then do;
      r4wks81=-3; r4ewks81=-3; end;
  if stopw>104 and self=1 and r4wks82>0 then do;
      r4wks82=-3; r4swks82=-3; end;
  if stopw>104 and self=0 and r4wks82>0 then do;
      r4wks82=-3; r4ewks82=-3; end;
  if stopw>156 and self=1 and r4wks83>0 then do;
      r4wks83=-3; r4swks83=-3; end;
  if stopw>156 and self=0 and r4wks83>0 then do;
      r4wks83=-3; r4ewks83=-3; end;
  if stopw>209 and self=1 and r4wks84>0 then do;
      r4wks84=-3; r4swks84=-3; end;
  if stopw>209 and self=0 and r4wks84>0 then do;
      r4wks84=-3; r4ewks4=-3; end;
  if stopw>261 and self=1 and r4wks85>0 then do;
      r4wks85=-3; r4swks85=-3; end;
  if stopw>261 and self=0 and r4wks85>0 then do;
      r4wks85=-3; r4ewks85=-3; end;
  if stopw>313 and self=1 and r4wks86>0 then do;
      r4wks86=-3; r4swks86=-3; end;
  if stopw>313 and self=0 and r4wks86>0 then do;
      r4wks86=-3; r4ewks86=-3; end;
  if stopw>365 and self=1 and r4wks87>0 then do;
      r4wks87=-3; r4swks87=-3; end;
  if stopw>365 and self=0 and r4wks87>0 then do;
      r4wks87=-3; r4ewks87=-3; end;
  if stopw>417 and self=1 and r4wks88>0 then do;
      r4wks88=-3; r4swks88=-3; end;
  if stopw>417 and self=0 and r4wks88>0 then do;
      r4wks88=-3; r4ewks88=-3; end;
  if stopw>470 and self=1 and r4wks89>0 then do;
      r4wks89=-3; r4swks89=-3; end;
  if stopw>470 and self=0 and r4wks89>0 then do;
      r4wks89=-3; r4ewks89=-3; end;
  if stopw>522 and self=1 and r4wks90>0 then do;
      r4wks90=-3; r4swks90=-3; end;
  if stopw>522 and self=0 and r4wks90>0 then do;
      r4wks90=-3; r4ewks90=-3; end;
  if stopw>574 and self=1 and r4wks91>0 then do;
      r4wks91=-3; r4swks91=-3; end;
  if stopw>574 and self=0 and r4wks91>0 then do;
      r4wks91=-3; r4ewks91=-3; end;
  if stopw>626 and self=1 and r4wks92>0 then do;
      r4wks92=-3; r4swks92=-3; end;
  if stopw>626 and self=0 and r4wks92>0 then do;
      r4wks92=-3; r4ewks92=-3; end;
  if stopw>678 and self=1 and r4wks93>0 then do;
      r4wks93=-3; r4swks93=-3; end;
  if stopw>678 and self=0 and r4wks93>0 then do;
      r4wks93=-3; r4ewks93=-3; end;
  if stopw>730 and self=1 and r4wks94>0 then do;
      r4wks94=-3; r4swks94=-3; end;
  if stopw>730 and self=0 and r4wks94>0 then do;
      r4wks94=-3; r4ewks94=-3; end;
  if stopw>783 and self=1 and r4wks95>0 then do;
      r4wks95=-3; r4swks95=-3; end;
  if stopw>783 and self=0 and r4wks95>0 then do;
      r4wks95=-3; r4ewks95=-3; end;
  if stopw>834 and self=1 and r4wks96>0 then do;
      r4wks96=-3; r4swks96=-3; end;
  if stopw>834 and self=0 and r4wks96>0 then do;
      r4wks96=-3; r4ewks96=-3; end;
  if stopw>887 and self=1 and r4wks97>0 then do;
      r4wks97=-3; r4swks97=-3; end;
  if stopw>887 and self=0 and r4wks97>0 then do;
      r4wks97=-3; r4ewks97=-3; end;
  if stopw>939 and self=1 and r4wks98>0 then do;
      r4wks98=-3; r4swks98=-3; end;
  if stopw>939 and self=0 and r4wks98>0 then do;
      r4wks98=-3; r4ewks98=-3; end;
  if stopw>990 and self=1 and r4wks99>0 then do;
      r4wks99=-3; r4swks99=-3; end;
  if stopw>990 and self=0 and r4wks99>0 then do;
      r4wks99=-3; r4ewks99=-3; end;
  if stopw>1043 and self=1 and r4wks00>0 then do;
      r4wks00=-3; r4swks00=-3; end;
  if stopw>1043 and self=0 and r4wks00>0 then do;
      r4wks00=-3; r4ewks00=-3; end;
  if stopw>1096 and self=1 and r4wks01>0 then do;
      r4wks01=-3; r4swks01=-3; end;
  if stopw>1096 and self=0 and r4wks01>0 then do;
      r4wks01=-3; r4ewks01=-3; end;
end;

/* stop date invalid  */
if stopfl=1 then do;
```

```
if starw<53 and self=1 and r4wks80>0 then do;
    r4wks80=-3; r4swks80=-3; end;
if starw<53 and self=0 and r4wks80>0 then do;
    r4wks80=-3; r4ewks80=-3; end;
if starw<105 and self=1 and r4wks81>0 then do;
    r4wks81=-3; r4swks81=-3;  end;
if starw<105 and self=0 and r4wks81>0 then do;
    r4wks81=-3; r4ewks81=-3; end;
if starw<157 and self=1 and r4wks82>0 then do;
    r4wks82=-3; r4swks82=-3; end;
if starw<157 and self=0 and r4wks82>0 then do;
    r4wks82=-3; r4ewks82=-3; end;
if starw<210 and self=1 and r4wks83>0 then do;
    r4wks83=-3; r4swks83=-3; end;
if starw<210 and self=0 and r4wks83>0 then do;
    r4wks83=-3; r4ewks83=-3; end;
if starw<262 and self=1 and r4wks84>0 then do;
    r4wks84=-3; r4swks84=-3; end;
if starw<262 and self=0 and r4wks84>0 then do;
    r4wks84=-3; r4ewks84=-3; end;
if starw<314 and self=1 and r4wks85>0 then do;
    r4wks85=-3; r4swks85=-3; end;
if starw<314 and self=0 and r4wks85>0 then do;
    r4wks85=-3; r4ewks85=-3; end;
if starw<366 and self=1 and r4wks86>0 then do;
    r4wks86=-3; r4swks86=-3; end;
if starw<366 and self=0 and r4wks86>0 then do;
    r4wks86=-3; r4ewks86=-3; end;
if starw<418 and self=1 and r4wks87>0 then do;
    r4wks87=-3; r4swks87=-3; end;
if starw<418 and self=0 and r4wks87>0 then do;
    r4wks87=-3; r4ewks87=-3; end;
if starw<471 and self=1 and r4wks88>0 then do;
    r4wks88=-3; r4swks88=-3; end;
if starw<471 and self=0 and r4wks88>0 then do;
    r4wks88=-3; r4ewks88=-3; end;
if starw<523 and self=1 and r4wks89>0 then do;
    r4wks89=-3; r4swks89=-3; end;
if starw<523 and self=0 and r4wks89>0 then do;
    r4wks89=-3; r4ewks89=-3; end;
if starw<575 and self=1 and r4wks90>0 then do;
    r4wks90=-3; r4swks90=-3; end;
if starw<575 and self=0 and r4wks90>0 then do;
    r4wks90=-3; r4ewks90=-3; end;
if starw<627 and self=1 and r4wks91>0 then do;
    r4wks91=-3; r4swks91=-3; end;
if starw<627 and self=0 and r4wks91>0 then do;
    r4wks91=-3; r4ewks91=-3; end;
if starw<679 and self=1 and r4wks92>0 then do;
    r4wks92=-3; r4swks92=-3; end;
if starw<679 and self=0 and r4wks92>0 then do;
    r4wks92=-3; r4ewks92=-3; end;
if starw<731 and self=1 and r4wks93>0 then do;
    r4wks93=-3; r4swks93=-3; end;
if starw<731 and self=0 and r4wks93>0 then do;
    r4wks93=-3; r4ewks93=-3; end;
if starw<784 and self=1 and r4wks94>0 then do;
    r4wks94=-3; r4swks94=-3; end;
if starw<784 and self=0 and r4wks94>0 then do;
    r4wks94=-3; r4ewks94=-3; end;
if starw<836 and self=1 and r4wks95>0 then do;
    r4wks95=-3; r4swks95=-3; end;
if starw<836 and self=0 and r4wks95>0 then do;
    r4wks95=-3; r4ewks95=-3; end;
if starw<888 and self=1 and r4wks96>0 then do;
    r4wks96=-3; r4swks96=-3; end;
if starw<888 and self=0 and r4wks96>0 then do;
    r4wks96=-3; r4ewks96=-3; end;
if starw<940 and self=1 and r4wks97>0 then do;
    r4wks97=-3; r4swks97=-3; end;
if starw<940 and self=0 and r4wks97>0 then do;
    r4wks97=-3; r4ewks97=-3; end;
if starw<991 and self=1 and r4wks98>0 then do;
    r4wks98=-3; r4swks98=-3; end;
if starw<991 and self=0 and r4wks98>0 then do;
    r4wks98=-3; r4ewks98=-3; end;
if starw<1044 and self=1 and r4wks99>0 then do;
    r4wks99=-3; r4swks99=-3; end;
if starw<1044 and self=0 and r4wks99>0 then do;
    r4wks99=-3; r4ewks99=-3; end;
if starw<1096 and self=1 and r4wks00>0 then do;
    r4wks00=-3; r4swks00=-3; end;
if starw<1096 and self=0 and r4wks00>0 then do;
    r4wks00=-3; r4ewks00=-3; end;
if starw<1149 and self=1 and r4wks01>0 then do;
    r4wks01=-3; r4swks01=-3; end;
if starw<1149 and self=0 and r4wks01>0 then do;
    r4wks01=-3; r4ewks01=-3; end;
end;
end;

*** Include valid skips;
do over r4wks;
  if e200=-5 then do;
    r4wks=-5; r4swks=-5; r4ewks=-5;
  end;
end;

/* Check that no start date (except in ignore cases) comes
before the R's dli. */
do over starw;
  if r2int=-5 and r3int=-5 and intwk>0 and starw>0 and
      starw<r1int and ignore ne 1 then why1=1;
  if r2int>0 and r3int=-5 and intwk>0 and starw>0 and
      starw<r2int and ignore ne 1 then why2=1;
  if r3int>0 and intwk>0 and starw>0 and starw<r3int
      and ignore ne 1 then why3=1;
end;

/* Check that -3's in c.v. come from right places. The
cases below should all have missing years. */
array smofl (i) smofl1-smofl9;
array emofl (i) emofl1-emofl9;
```

```
do over starw;
  if (r4wks98=-3 or r4wks99=-3 or r4wks00=-3 or
      r4wks01=-3) then do;
    if smofl=0 and emofl=0 and bgfl1_1=0 & bgfl1_2=0
      & bgfl1_3=0 & bgfl1_4=0 & bgfl1_5=0 &
      bgfl1_6=0 & bgfl1_7=0 & bgfl1_8=0 & bgfl1_9=0
      & bgfl1_10=0 & bgfl2_1=0 & bgfl2_2=0 &
      bgfl2_3=0 & bgfl2_4=0 & bgfl2_5=0 & bgfl3_1=0
      & bgfl3_2=0 & bgfl3_3=0 & bgfl4_1=0 &
      bgfl4_2=0 & bgfl4_3=0 & bgfl5_1=0 & bgfl6_1=0
      & egfl1_1=0 & egfl1_2=0 & egfl1_3=0 &
      egfl1_4=0 & egfl1_5=0 & egfl1_6=0 & egfl1_7=0
      & egfl1_8=0 & egfl1_9=0 & egfl1_10=0 &
      egfl2_1=0 & egfl2_2=0 & egfl2_3=0 & egfl2_4=0
      & egfl2_5=0 & egfl3_1=0 & egfl3_2=0 &
      egfl3_3=0 & egfl4_1=0 & egfl4_2=0 & egfl4_3=0
      & egfl5_1=0 & egfl6_1=0
    then yearmiss=1;
  end;
end;

/* Round 3 variable */
array r3wks r3wks80-r3wks99 r3wks00;
/* Total created variable */
array twks twks80-twks99 twks00;

do over twks; r3wks=twks; end;

/* Define the following c.v.'s to match up the array sizes,
they are zero by definition. */
r1wks99=0; r1wks00=0; r1wks01=0;
r2wks00=0; r2wks01=0;
r3wks01=0;

/* Round 1 created variable */
array r1wks (i) r1wks80-r1wks99 r1wks00 r1wks01;
/* Round 2 created variable */
array r2wks (i) r2wks80-r2wks99 r2wks00 r2wks01;
/* Round 3 created variable */
array r3wks (i) r3wks80-r3wks99 r3wks00 r3wks01;
/* Round 4 created variable, all jobs */
array r4wks (i) r4wks80-r4wks99 r4wks00 r4wks01;
/* Round 4 created variable, self-employed jobs */
array r4swks (i) r4swks80-r4swks99 r4swks00 r4swks01;
/* Round 4 created variable, employee jobs */
array r4ewks (i) r4ewks80-r4ewks99 r4ewks00 r4ewks01;
/* Total created variable, all jobs */
array twks (i) twks80-twks99 twks00 twks01;
/* Total created variable, self-employed jobs */
array tswks (i) tswks80-tswks99 tswks00 tswks01;
/* Total created variable, employee jobs  */
array tewks (i) tewks80-tewks99 tewks00 tewks01;

/* Begin by splitting up periods where Round4 the dli c.v.
exclusively collect weeks worked information. Define the
dli interview year as the split. Any weeks worked
information collected before the dli interview week
should be independent of information collected in
Round4. */
/* For the new Round 4 c.v.'s, we are assuming that all dli
jobs are employee-type jobs. Thus, dli information will be
added to the "all jobs" and "employee jobs" c.v.'s but not
"self-employed jobs" c.v.'s. */

do over r4wks;
  twks=0; tswks=0; tewks=0;
  if r4wks>0 then do; twks=r4wks; end;
  if r4swks>0 then do; tswks=r4swks; end;
  if r4ewks>0 then do; tewks=r4ewks; end;
  if r3wks>0 and dliwk=r3int then do; twks=r3wks;
      tewks=r3wks; end;
  if r2wks>0 and dliwk=r2int then do; twks=r2wks;
      tewks=r2wks; end;
  if r1wks>0 and dliwk=r1int then do; twks=r1wks;
      tewks=r1wks; end;
  if r3wks>0 and r4wks>0 and dliwk=r3int then do;
      twks=r3wks+r4wks; end;
  if r3wks>0 and r4ewks>0 and dliwk=r3int then do;
      tewks=r3wks+r4ewks; end;
  if r2wks>0 and r4wks>0 and dliwk=r2int then do;
      twks=r2wks+r4wks; end;
  if r2wks>0 and r4ewks>0 and dliwk=r2int then do;
      tewks=r2wks+r4ewks; end;
  if r1wks>0 and r4wks>0 and dliwk=r1int then do;
      twks=r1wks+r4wks; end;
  if r1wks>0 and r4ewks>0 and dliwk=r1int then do;
      tewks=r1wks+r4ewks; end;
end;

/* If either created variable from Round 4 or dli is a -2 or
-3, then the overall created variable will be -2 or -3. */
do over r4wks;
  if r4wks in (-1,-2,-3) then do; twks=-3; end;
  if r4swks in (-1,-2,-3) then do; tswks=-3; end;
  if r4ewks in (-1,-2,-3) then do; tewks=-3; end;
  if r3wks in (-1,-2,-3) and dliwk=r3int then do; twks=-3;
      tewks=-3; end;
  if r2wks in (-1,-2,-3) and dliwk=r2int then do; twks=-3;
      tewks=-3; end;
  if r1wks in (-1,-2,-3) and dliwk=r1int then do; twks=-3;
      tewks=-3; end;
end;

/* Round 4 non-interview cases */
do over twks;
  if intwk=-5 then do; twks=-5; tswks=-5; tewks=-5; end;
end;

endsas;
```

NUMBER OF WEEKS WORKED SINCE LAST INTERVIEW

Variables Created: CV_WKSWK_DLI_ALL
CV_WKSWK_DLI_ET
CV_WKSWK_DLI_SE

Programs Used

This program uses **emp_begin.sas** as input (see the first page of this appendix for details).

For each individual, this program counts the number of weeks the respondent worked since the last interview. Three variables are created: one for all jobs, one for employee-type jobs, and one for self-employed jobs. **See the introduction to this appendix for important information about the universes for each of these variables.** Respondents not working in a given year are given a default value of zero.

```
array job1wks (i) wk1_1-wk1_1148;
array job2wks (i) wk2_1-wk2_1148;
array job3wks (i) wk3_1-wk3_1148;
array job4wks (i) wk4_1-wk4_1148;
array job5wks (i) wk5_1-wk5_1148;
array job6wks (i) wk6_1-wk6_1148;
array job7wks (i) wk7_1-wk7_1148;
array job8wks (i) wk8_1-wk8_1148;
array job9wks (i) wk9_1-wk9_1148;
array alljobs (i) wks1-wks1148;
array selfjobs (i) swks1-swks1148;
array empjobs (i) ewks1-ewks1148;
array starw (i) starw1-starw9;
array stopw (i) stopw1-stopw9;
array self (i) self1-self9;

/* Create separate arrays for self employed jobs and
employee jobs. The arrays are laid down in the same
manner as "alljobs," with the self-employed dummy
deciding which jobs are laid in each array. */
do over alljobs; alljobs=0; selfjobs=0; empjobs=0; end;

/** Overlay multiple jobs over JOB 1 work weeks      **/
do over job1wks;
  alljobs=job1wks;
  if self1=1 then do; selfjobs=job1wks; end;
  if self1=0 then do; empjobs=job1wks; end;
end;

do over alljobs;
  if job2wks=1 then do; alljobs=job2wks; end;
  if job2wks=1 & self2=1 then do; selfjobs=job2wks; end;
  if job2wks=1 & self2=0 then do; empjobs=job2wks;
      end;
  if job2wks=-3 & alljobs=0 then do; alljobs=job2wks;
      end;
  if job2wks=-3 & selfjobs=0 and self2=1 then do;
      selfjobs=job2wks; end;
  if job2wks=-3 & empjobs=0 and self2=0 then do;
      empjobs=job2wks; end;
end;

/*At this point the program loops through the job2 code
for jobs 3-9. Contact NLS User Services for details.*/

/** Calculate cumulative weeks on all jobs since dli **/
allwks=0; selfwks=0; empwks=0;

if dliwk>0 then do;
  if dliwk=>age14wk then do;
    do i=dliwk to 1148;
      if alljobs=1 then do; allwks=allwks+1; end;
      if selfjobs=1 then do; selfwks=selfwks+1; end;
      if empjobs=1 then do; empwks=empwks+1; end;
    end;
    do i=dliwk to 1148;
      if alljobs=-3 then do; allwks=-3; end;
      if selfjobs=-3 then do; selfwks=-3; end;
      if empjobs=-3 then do; empwks=-3; end;
    end;
  end;
  if age14wk>dliwk then do;
    do i=age14wk to 1148;
      if alljobs=1 then do; allwks=allwks+1; end;
      if selfjobs=1 then do; selfwks=selfwks+1; end;
      if empjobs=1 then do; empwks=empwks+1; end;
    end;
    do i=age14wk to 1148;
      if alljobs=-3 then do; allwks=-3; end;
      if selfjobs=-3 then do; selfwks=-3; end;
      if empjobs=-3 then do; empwks=-3; end;
    end;
  end;
end;

if starfl_1=1 then do; allwks=-3; end;
/*repeat for starfl_2 through starfl_9*/
if stopfl_1=1 then do; allwks=-3; end;
/*repeat for stopfl_2 through stopfl_9*/

if e200=-5 then do; allwks=-5; selfwks=-5; empwks=-5;
end;
endsas;
```

NUMBER OF WEEKS WORKED AGE 14–19, WEEKS WORKED AT ALL JOBS SINCE AGE 20, AND WEEKS WORKED AT EMPLOYEE-TYPE JOBS SINCE AGE 20

Variables Created: CV_WKSWK_TEEN
CV_WKSWK_ADULT_ET
CV_WKSWK_ADULT_ALL

Programs Used

This program uses **emp_begin.sas** as input (see the first page of this appendix for details).

For each respondent, this program creates a variable counting the number of weeks worked at any employee-type job between the ages of 14 and 19. It then creates two variables counting weeks worked from age 20 to the present; one totals all jobs and one presents only employee jobs. **See the introduction to this appendix for important information about the universes for each of these variables.** Respondents not working are given a default value of zero; respondents who have not yet reached age 20 receive a valid skip (-4) on the latter two variables.

```
array job1wks (i) wk1_1-wk1_1148;
array job2wks (i) wk2_1-wk2_1148;
array job3wks (i) wk3_1-wk3_1148;
array job4wks (i) wk4_1-wk4_1148;
array job5wks (i) wk5_1-wk5_1148;
array job6wks (i) wk6_1-wk6_1148;
array job7wks (i) wk7_1-wk7_1148;
array job8wks (i) wk8_1-wk8_1148;
array job9wks (i) wk9_1-wk9_1148;
array alljobs (i) wks1-wks1148;
array empjobs (i) ewks1-ewks1148;
array starw (i) starw1-starw9;
array stopw (i) stopw1-stopw9;
array starfl (i) starfl_1-starfl_9;
array stopfl (i) stopfl_1-stopfl_9;

/** Overlay multiple jobs over JOB 1 work weeks      **/

do over empjobs; empjobs=0; end;
do over job1wks;
  if self1=0 then do; empjobs=job1wks; end;
end;

do over empjobs;
  if job2wks=1 and self2=0 then do;
    empjobs=job2wks;
  end;
  if job2wks=-3 and self2=0 and empjobs=0 then do;
    empjobs=job2wks;
  end;
end;

/*Repeat the job 2 code above for jobs 3-9; contact NLS
User Services for more information.*/

do over alljobs; alljobs=0; end;
do over job1wks; alljobs=job1wks; end;

do over alljobs;
  if job2wks=1 then do;
    alljobs=job2wks;
  end;
  if job2wks=-3 and alljobs=0 then do;
    alljobs=job2wks;
  end;
end;

/*Repeat the job 2 code above for jobs 3-9; contact NLS
User Services for more information.*/

/*Calculate cumulative weeks on all jobs between age 14
and age 19, and on all jobs and employee jobs since age
20*/
/* r4emp14 is the variable that counts the number of
weeks worked since respondent's 14th birthday. This is
changed from wks14 to distinguish it from a member in
the alljobs array. */
r4emp14=0;
r4wks20=0;
r4emp20=0;

/* Note the max for age20wk is 1305. */
if age14wk>0 and (age20wk-1)<=1148 then do;
  do i=age14wk to (age20wk-1);
    if empjobs=1 then do; r4emp14=r4emp14+1; end;
    if empjobs=-3 then do; r4emp14=-3; end;
  end;
end;

if age14wk>0 and (age20wk-1)>1148 then do;
  do i=age14wk to 1148;
    if empjobs=1 then do; r4emp14=r4emp14+1; end;
    if empjobs=-3 then do; r4emp14=-3; end;
  end;
end;

if age14wk>0 and age20wk<=1148 then do;
  do i=age20wk to 1148;
    if alljobs=1 then do; r4wks20=r4wks20+1; end;
    if alljobs=-3 then do; r4wks20=-3; end;
```

```
   end;
end;

if age14wk>0 and age20wk<=1148 then do;
   do i=age20wk to 1148;
      if empjobs=1 then do; r4emp20=r4emp20+1; end;
      if empjobs=-3 then do; r4emp20=-3; end;
   end;
end;

/* lowest age14wk is 731, lowest age20wk is 1044*/
do i=1 to 9;
if starfl=1 and r4emp14>0 then do;
 if age14wk<784 then do;
   if stopw>730 then do; r4emp14=-3; end;
 end;
 if age14wk<836 then do;
   if stopw>783 then do; r4emp14=-3; end;
 end;
 if age14wk<888 then do;
   if stopw>835 then do; r4emp14=-3; end;
 end;
 if age14wk<940 then do;
   if stopw>887 then do; r4emp14=-3; end;
 end;
 if age14wk<991 then do;
   if stopw>939 then do; r4emp14=-3; end;
 end;
 if age14wk<1044 then do;
   if stopw>990 then do; r4emp14=-3; end;
 end;
 if age14wk<1097 then do;
   if stopw>1043 then do; r4emp14=-3; end;
 end;
 if age14wk<1149 then do;
   if stopw>1096 then do; r4emp14=-3; end;
 end;
end;

if starfl=1 and r4wks20>0 then do;
 if age20wk<1097 then do;
   if stopw>1043 then do; r4wks20=-3; r4emp20=-3; end;
 end;
 if age20wk<1149 then do;
   if stopw>1096 then do; r4wks20=-3; r4emp20=-3; end;
 end;
end;

if stopfl=1 and r4emp14>0 then do;
 if age14wk<784 then do;
   if starw<784 then do; r4emp14=-3; end;
 end;
 if age14wk<836 then do;
   if starw<836 then do; r4emp14=-3; end;
 end;
 if age14wk<888 then do;
   if starw<888 then do; r4emp14=-3; end;
 end;
 if age14wk<940 then do;
   if starw<940 then do; r4emp14=-3; end;
 end;
 if age14wk<991 then do;
   if starw<991 then do; r4emp14=-3; end;
 end;
 if age14wk<1044 then do;
   if starw<1044 then do; r4emp14=-3; end;
 end;
 if age14wk<1097 then do;
   if starw<1097 then do; r4emp14=-3; end;
 end;
 if age14wk<1149 then do;
   if starw<1149 then do; r4emp14=-3; end;
 end;
end;

if stopfl=1 and r4wks20>0then do;
 if age20wk<1097 then do;
   if starw<1097 then do; r4wks20=-3; r4emp20=-3; end;
 end;
 if age20wk<1149 then do;
   if starw<1149 then do; r4wks20=-3; r4emp20=-3; end;
 end;
end;
end;

*** Include noninterviews;
if e200=-5 then do;
   r4emp14=-5; r4wks20=-5;  r4emp20=-3;
end;

/* This program calculates the total number of weeks
worked since age 14. Since the tenure created variables
from Round 4 and dli Round are only calculated during
the interview period, this program sums up these tenure
dates to get a total weeks count. */

r3wks14=totwks14;
r3wks20=totwks20;
drop totwks14 totwks20;

/* Initialize created variable for both rounds to zero. */
temp14=0;
twks20=0;
temp20=0;

/* By the construction of the Round 4 created variable, we
can simply add the two created variables from Round 4
and dli Round if they are both postive. If one is positive
and one is zero, then the positive value will be the total
created variable for both rounds. If neither is positive,
then the total created variable for both rounds will be zero.
If either the Round 4 c.v. or the dli c.v. is invalid, then the
total c.v. will be -3. */
```

```
if r4emp14=>0 then do; temp14=r4emp14; end;
if r4emp14=-3 then do; temp14=-3; end;

/* dli is Round 3 */
if r3wks14=>0 and r4emp14=>0 then do;
  temp14=r3wks14+r4emp14;
end;
if r3wks14=-3 then do; temp14=-3; end;

/* dli is Round 2 */
if r2wks14=>0 and r4emp14=>0 and r3int=-5 then do;
  temp14=r2wks14+r4emp14;
end;
if r2wks14=-3 and r3int=-5 then do; temp14=-3; end;

/* dli is Round 1 */
if r1wks14=>0 and r4emp14=>0 and r3int=-5 and
r2int=-5 then do;
  temp14=r1wks14+r4emp14;
end;
if r1wks14=-3 and r3int=-5 and r2int=-5 then do;
  temp14=-3;
end;

if r4wks20>0 then twks20=r4wks20;
if r3wks20>0 then twks20=r3wks20;
if r4wks20>0 and r3wks20>0 then do;
  twks20=r4wks20+r3wks20;
end;
if r4wks20=-3 or r3wks20=-3 then do;
  twks20=-3;
end;

if r4emp20>0 then temp20=r4emp20;
if r3wks20>0 then temp20=r3wks20;
if r4emp20>0 and r3wks20>0 then do;
  temp20=r4emp20+r3wks20;
end;
if r4emp20=-3 or r3wks20=-3 then do;
  temp20=-3;
end;

if uid1=-5 then do;
  temp14=-5; twks20=-5; temp20=-5;
end;

/* For respondents younger than 20 at interview date,
their thrs14 c.v. will be -4. */
if age20wk>intwk and intwk ne -5 then do;
  twks20=-4; temp20=-4;
end;

endsas;
```

WEEKS WORKED AT EMPLOYEE JOB #X DURING 19XX/20XX

Variables Created: CV_WKSWK_JOB_YR.01.80 – CV_WKSWK_JOB_YR.01.01
CV_WKSWK_JOB_YR.02.80 – CV_WKSWK_JOB_YR.02.01 etc. through job #9

Programs Used

This program uses **emp_begin.sas** as input (see the first page of this appendix for details).

This program creates variables for each of the respondent's jobs counting the number of weeks worked in each calendar year. A variable is created for each potential job even if the respondent has worked no jobs in a given year with the default value set to zero (0). The most jobs held by any respondent in round 4 was nine, so variables are created for nine jobs for each respondent.

```
/* Section 1: Create the Round3 variable for number of
weeks worked at a given job in a given year. */

array starw (i) starw1-starw9;
array stopw (i) stopw1-stopw9;
array job1wks (i) wk1_1-wk1_1148;
/*repeat for jobs 2-9 (e.g. job2wks = wk2_1-wk2_1148)*/
array starfl (i) starfl_1-starfl_9;
array stopfl (i) stopfl_1-stopfl_9;
array r4wks80 (i) r4wks801-r4wks809;
array r4wks81 (i) r4wks811-r4wks819;
  /*and so on through*/
array r4wks01 (i) r4wks011-r4wks019;

/** Calculate cumulative weeks on individual jobs for
each year **/

/* 1980 */

r4wks801=0; r4wks802=0; r4wks803=0; r4wks804=0;
r4wks805=0; r4wks806=0; r4wks807=0; r4wks808=0;
r4wks809=0;

do i=1 to 52;
  if job1wks=1 then do; r4wks801=r4wks801+1; end;
  if job2wks=1 then do; r4wks802=r4wks802+1; end;
  if job3wks=1 then do; r4wks803=r4wks803+1; end;
  if job4wks=1 then do; r4wks804=r4wks804+1; end;
  if job5wks=1 then do; r4wks805=r4wks805+1; end;
  if job6wks=1 then do; r4wks806=r4wks806+1; end;
  if job7wks=1 then do; r4wks807=r4wks807+1; end;
  if job8wks=1 then do; r4wks808=r4wks808+1; end;
  if job9wks=1 then do; r4wks809=r4wks809+1; end;

  if job1wks=-3  then do; r4wks801=-3; end;
  if job2wks=-3 then do; r4wks802=-3; end;
  if job3wks=-3 then do; r4wks803=-3; end;
  if job4wks=-3 then do; r4wks804=-3; end;
  if job5wks=-3 then do; r4wks805=-3; end;
  if job6wks=-3 then do; r4wks806=-3; end;
  if job7wks=-3 then do; r4wks807=-3; end;
  if job8wks=-3 then do; r4wks808=-3; end;
  if job9wks=-3 then do; r4wks809=-3; end;
end;

/*At this point the program repeats the above code for
each year through 2001. Due to space considerations this
code is not included here. Researchers needing the
complete code should contact NLS User Services. The
variables and "do i" statements for each year are as follows:
    1981    wksw811-wksw819    do i=53 to 104
    1982    wksw821-wksw829    do i=105 to 156
    1983    wksw831-wksw839    do i=157 to 209
    1984    wksw841-wksw849    do i=210 to 261
    1985    wksw851-wksw859    do i=262 to 313
    1986    wksw861-wksw869    do i=314 to 365
    1987    wksw871-wksw879    do i=366 to 417
    1988    wksw881-wksw889    do i=418 to 470
    1989    wksw891-wksw899    do i=471 to 522
    1990    wksw901-wksw909    do i=523 to 574
    1991    wksw911-wksw919    do i=575 to 626
    1992    wksw921-wksw929    do i=627 to 678
    1993    wksw931-wksw939    do i=679 to 730
    1994    wksw941-wksw949    do i=731 to 783
    1995    wksw951-wksw959    do i=784 to 835
    1996    wksw961-wksw969    do i=836 to 887
    1997    wksw971-wksw979    do i=888 to 939
    1998    wksw981-wksw989    do i=940 to 991
    1999    wksw991-wksw999    do i=992 to 1044
    2000    wksw001-wksw009    do i=1045 to 1098   **/
    2000    wksw011-wksw019    do i=1097 to 1148   **/

do i=1 to 9;
/*start date invalid*/
if starfl=1 then do;
  if r4wks80>0 & stopw>1 then do; r4wks80=-3; end;
  if r4wks81>0 & stopw>52 then do; r4wks81=-3; end;
  if r4wks82>0 & stopw>104 then do; r4wks82=-3; end;
  if r4wks83>0 & stopw>156 then do; r4wks83=-3; end;
  if r4wks84>0 & stopw>209 then do; r4wks84=-3; end;
  if r4wks85>0 & stopw>261 then do; r4wks85=-3; end;
  if r4wks86>0 & stopw>313 then do; r4wks86=-3; end;
  if r4wks87>0 & stopw>365 then do; r4wks87=-3; end;
  if r4wks88>0 & stopw>417 then do; r4wks88=-3; end;
  if r4wks89>0 & stopw>470 then do; r4wks89=-3; end;
  if r4wks90>0 & stopw>522 then do; r4wks90=-3; end;
  if r4wks91>0 & stopw>574 then do; r4wks91=-3; end;
```

```
  if r4wks92>0 & stopw>626 then do; r4wks92=-3; end;
  if r4wks93>0 & stopw>678 then do; r4wks93=-3; end;
  if r4wks94>0 & stopw>730 then do; r4wks94=-3; end;
  if r4wks95>0 & stopw>783 then do; r4wks95=-3; end;
  if r4wks96>0 & stopw>835 then do; r4wks96=-3; end;
  if r4wks97>0 & stopw>887 then do; r4wks97=-3; end;
  if r4wks98>0 & stopw>939 then do; r4wks98=-3; end;
  if r4wks99>0 & stopw>991 then do; r4wks99=-3; end;
  if r4wks00>0 & stopw>1043 then do; r4wks00=-3; end;
  if r4wks01>0 & stopw>1096 then do; r4wks01=-3; end;
end;

/* stop date invalid */
if stopfl=1 then do;
  if r4wks80>0 & starw<53 then do; r4wks80=-3; end;
  if r4wks81>0 & starw<105 then do; r4wks81=-3; end;
  if r4wks82>0 & starw<157 then do; r4wks82=-3; end;
  if r4wks83>0 & starw<210 then do; r4wks83=-3; end;
  if r4wks84>0 & starw<262 then do; r4wks84=-3; end;
  if r4wks85>0 & starw<314 then do; r4wks85=-3; end;
  if r4wks86>0 & starw<366 then do; r4wks86=-3; end;
  if r4wks87>0 & starw<418 then do; r4wks87=-3; end;
  if r4wks88>0 & starw<471 then do; r4wks88=-3; end;
  if r4wks89>0 & starw<523 then do; r4wks89=-3; end;
  if r4wks90>0 & starw<575 then do; r4wks90=-3; end;
  if r4wks91>0 & starw<627 then do; r4wks91=-3; end;
  if r4wks92>0 & starw<679 then do; r4wks92=-3; end;
  if r4wks93>0 & starw<731 then do; r4wks93=-3; end;
  if r4wks94>0 & starw<784 then do; r4wks94=-3; end;
  if r4wks95>0 & starw<836 then do; r4wks95=-3; end;
  if r4wks96>0 & starw<888 then do; r4wks96=-3; end;
  if r4wks97>0 & starw<940 then do; r4wks97=-3; end;
  if r4wks98>0 & starw<991 then do; r4wks98=-3; end;
  if r4wks99>0 & starw<1044 then do; r4wks99=-3; end;
  if r4wks00>0 & starw<1097 then do; r4wks00=-3; end;
  if r4wks01>0 & starw<1149 then do; r4wks01=-3; end;
end;
end;

if e200=-5 then do;
   r4wks801=-5; r4wks802=-5; r4wks803=-5; r4wks804=-
        5; r4wks805=-5; r4wks806=-5; r4wks807=-5;
        r4wks808=-5; r4wks809=-5;
   /* and so on through*/
   r4wks011=-5; r4wks012=-5; r4wks013=-5; r4wks014=-
        5; r4wks015=-5; r4wks016=-5; r4wks017=-5;
        r4wks018=-5; r4wks019=-5;
end;

/*Now merge in the information from previous rounds*/
/* Round 3 variable */
array r3wks80 r3wks801-r3wks809;
array r3wks81 r3wks811-r3wks819;
array r3wks82 r3wks821-r3wks829;
array r3wks83 r3wks831-r3wks839;
array r3wks84 r3wks841-r3wks849;
array r3wks85 r3wks851-r3wks859;
array r3wks86 r3wks861-r3wks869;
array r3wks87 r3wks871-r3wks879;
array r3wks88 r3wks881-r3wks889;
array r3wks89 r3wks891-r3wks899;
array r3wks90 r3wks901-r3wks909;
array r3wks91 r3wks911-r3wks919;
array r3wks92 r3wks921-r3wks929;
array r3wks93 r3wks931-r3wks939;
array r3wks94 r3wks941-r3wks949;
array r3wks95 r3wks951-r3wks959;
array r3wks96 r3wks961-r3wks969;
array r3wks97 r3wks971-r3wks979;
array r3wks98 r3wks981-r3wks989;
array r3wks99 r3wks991-r3wks999;
array r3wks00 r3wks001-r3wks009;

/* Total created variable */
array twks80 twks801-twks809;
array twks81 twks811-twks819;
array twks82 twks821-twks829;
array twks83 twks831-twks839;
array twks84 twks841-twks849;
array twks85 twks851-twks859;
array twks86 twks861-twks869;
array twks87 twks871-twks879;
array twks88 twks881-twks889;
array twks89 twks891-twks899;
array twks90 twks901-twks909;
array twks91 twks911-twks919;
array twks92 twks921-twks929;
array twks93 twks931-twks939;
array twks94 twks941-twks949;
array twks95 twks951-twks959;
array twks96 twks961-twks969;
array twks97 twks971-twks979;
array twks98 twks981-twks989;
array twks99 twks991-twks999;
array twks00 twks001-twks009;

do over twks80;
   r3wks80=twks80;          r3wks81=twks81;
   r3wks82=twks82;          r3wks83=twks83;
   r3wks84=twks84;          r3wks85=twks85;
   r3wks86=twks86;          r3wks87=twks87;
   r3wks88=twks88;          r3wks89=twks89;
   r3wks90=twks90;          r3wks91=twks91;
   r3wks92=twks92;          r3wks93=twks93;
   r3wks94=twks94;          r3wks95=twks95;
   r3wks96=twks96;          r3wks97=twks97;
   r3wks98=twks98;          r3wks99=twks99;
   r3wks00=twks00;
end;
/* WKS945 is the Round 2 created variable for 5th job
(i.e. 5th UID code) in 1994. WKSW945 is the Round 3
created varible of the same description. r1wks594 is the
variable that combines these r1wo created variables (of the
same description). These infile variable names are
```

```
confusing, so rename them to more familiar "r1wks,
r2wks" etc. r1wks remains the total created variable. */

/* Note there are 7 jobs in Round 1, array matched to
make programming easier */
array r1uid (i) r1uid1-r1uid9;
array r2uid (i) r2uid1-r2uid9;
array r3uid (i) r3uid1-r3uid9;
array r4uid (i) uid1-uid9;

/* Arrange the jobs by year for Round 1*/
array r1wks80 (i) r1wks801-r1wks809;
array r1wks81 (i) r1wks811-r1wks819;
  /*and so on through*/
array r1wks99 (i) r1wks991-r1wks999;

/* Arrange the jobs by year for Round 2*/
array r2wks80 (i) r2wks801-r2wks809;
array r2wks81 (i) r2wks811-r2wks819;
  /*and so on through*/
array r2wks00 (i) r2wks001-r2wks009;

/* Arrange the jobs by year for Round 3*/
array r3wks80 (i) r3wks801-r3wks809;
array r3wks81 (i) r3wks811-r3wks819;
  /*and so on through*/
array r3wks00 (i) r3wks001-r3wks009;

/* Arrange the jobs by year for Round 4*/
array r4wks80 (i) r4wks801-r4wks809;
  /*and so on through*/
array r4wks01 (i) r4wks011-r4wks019;

/* Arrange the jobs by year for total c.v. */
array twks80 (i) twks801-twks809;
array twks81 (i) twks811-twks819;
  /*and so on through*/
array twks01 (i) twks011-twks019;

/* Initialize total weeks count to zero */
do i=1 to 9;
  twks80=0; twks81=0; twks82=0; twks83=0; twks84=0;
      twks85=0; twks86=0; twks87=0; twks88=0;
      twks89=0; twks90=0; twks91=0; twks92=0;
      twks93=0; twks94=0; twks95=0; twks96=0;
      twks97=0; twks98=0; twks99=0; twks00=0;
      twks01=0;
end;

/* Accounts for Round 3 non-interview cases */
do i=1 to 9;
  if e200=-5 then do;
    twks80=-5; twks81=-5; twks82=-5; twks83=-5;
      twks84=-5; twks85=-5; twks86=-5; twks87=-5;
      twks88=-5; twks89=-5; twks90=-5; twks91=-5;
      twks92=-5; twks93=-5; twks94=-5; twks95=-5;
      twks96=-5; twks97=-5; twks98=-5; twks99=-5;
      twks00=-5; twks01=-5;
  end;
end;

/* Begin by updating the total created variable if the
      Round 4 created variable is positive. For this
      variable, we are only worried about Round 4 jobs
      and previous round jobs that were worked in Round
      4. If a invalid negative value is given (don't know,
      refusal, or non-interview) then total created variable
      will be that negative value. */

/* 1980 */
do i=1 to 9;
  if r4wks80=>0 then do; twks80=r4wks80; end;
  if -4<r4wks80<0 then do; twks80=-3; end;
end;

/* 1981 */
do i=1 to 9;
  if r4wks81=>0 then do; twks81=r4wks81; end;
  if -4<r4wks81<0 then do; twks81=-3; end;
end;

/*and so on through*/
/* 1996 */
do i=1 to 9;
  if r4wks96=>0 then do; twks96=r4wks96; end;
  if -4<r4wks96<0 then do; twks96=-3; end;
end;

/* Since 1997-2000 are the interview years, we need to
      account for both Round 4 and dli created variables.
      Begin with negative conditions for the R4 c.v.*/

/* 1997 */
do i=1 to 9;
  if -4<r4wks97<0 then do; twks97=-3; end;
  if r4wks97=>0 then do; twks97=r4wks97; end;
end;

/* 1998 */
do i=1 to 9;
  if -4<r4wks98<0 then do; twks98=-3; end;
  if r4wks98=>0 then do; twks98=r4wks98; end;
end;

/* 1999 */
do i=1 to 9;
  if -4<r4wks99<0 then do; twks99=-3; end;
  if r4wks99=>0 then do; twks99=r4wks99; end;
end;

/* 2000 */
do i=1 to 9;
  if -4<r4wks00<0 then do; twks00=-3; end;
```

```
  if r4wks00>0 then do; twks00=r4wks00; end;
end;

/* 2001 */
do i=1 to 9;
  if -4<r4wks01<0 then do; twks01=-3; end;
  if r4wks01>0 then do; twks01=r4wks01; end;
end;

array match1 (i) match11-match19;
array match2 (i) match21-match29;
  /*and so on through*/
array match9 (i) match91-match99;

do i=1 to 9;
  match1=0; match2=0; match3=0; match4=0;
      match5=0; match6=0; match7=0; match8=0;
      match9=0;
end;

/* Define match14 as the dummy variable that equals one
      when the first job on the R1 UID roster and the
      fourth job on the R2 roster have the same UID. */

/* If any UID from the first position in a previous round
      matches with any UID in Round 4.  Need to be
      sure to find the most recent round that matches. */
do over r4uid;
  if r3uid1>0 and r4uid>0 and r3int>0 and intwk>0 then
      do;
    if r3uid1=r4uid then do; match1=3; end;
  end;
  if r2uid1>0 and r4uid>0 and match1=0 and intwk>0
      then do;
    if r2uid1=r4uid then do; match1=2; end;
  end;
  if r1uid1>0 and r4uid>0 and match1=0 and intwk>0
      then do;
    if r1uid1=r4uid then do; match1=1; end;
  end;
end;

/* Match UID from the second position in previous
      rounds to UIDs in Round 4 */
do over r4uid;
  if r3uid2>0 and r4uid>0 and r3int>0 and intwk>0 then
      do;
    if r3uid2=r4uid then do; match2=3; end;
  end;
  if r2uid2>0 and r4uid>0 and match2=0 and intwk>0
      then do;
    if r2uid2=r4uid then do; match2=2; end;
  end;
  if r1uid2>0 and r4uid>0 and match2=0 and intwk>0
      then do;
    if r1uid2=r4uid then do; match2=1; end;
  end;
end;

/*repeat the above code for UIDs in positions 3-9 in
      previous rounds*/

r4match_1=max(match11,match21,match31,match41,
      match51,match61,match71,match81,match91);
r4match_2=max(match12,match22,match32,match42,
      match52,match62,match72,match82,match92);
  /* and so on through */
r4match_9=max(match19,match29,match39,match49,
      match59,match69,match79,match89,match99);


if match11<r4match_1 then match11=0;
if match21<r4match_1 then match21=0;
  /* and so on through */
if match91<r4match_1 then match91=0;

if match12<r4match_2 then match12=0;
if match22<r4match_2 then match22=0;
  /* and so on through */
if match92<r4match_2 then match92=0;

if match13<r4match_3 then match13=0;
if match23<r4match_3 then match23=0;
  /* and so on through */
if match93<r4match_3 then match93=0;

if match14<r4match_4 then match14=0;
if match24<r4match_4 then match24=0;
  /* and so on through */
if match94<r4match_4 then match94=0;

if match15<r4match_5 then match15=0;
if match25<r4match_5 then match25=0;
  /* and so on through */
if match95<r4match_5 then match95=0;

if match16<r4match_6 then match16=0;
if match26<r4match_6 then match26=0;
  /* and so on through */
if match96<r4match_6 then match96=0;

if match17<r4match_7 then match17=0;
if match27<r4match_7 then match27=0;
  /* and so on through */

if match18<r4match_8 then match18=0;
if match28<r4match_8 then match28=0;
  /* and so on through */
if match98<r4match_8 then match98=0;

if match19<r4match_9 then match19=0;
if match29<r4match_9 then match29=0;
  /* and so on through */
if match99<r4match_9 then match99=0;
```

```
/* Now reassign total created variable if there is an
      identical job was worked in both rounds. Using the
      match variable, a job worked in both rounds will
      update the total created variable. Need to add
      positivity condition on the weeks worked values
      since some of the values equal -3. */

/* For respondents last interviewed in Round 3, 2000 */
if r3int_y=2000 then do;
  do over match1;
    if match1=3 then do;
      if r3wks801 not in (-5,-4,0,.) then
          twks80=r3wks801;
      if r3wks811 not in (-5,-4,0,.) then
          twks81=r3wks811;
        /*and so on through*/
      if r3wks991 not in (-5,-4,0,.) then
          twks99=r3wks991;
    end;
  end;

  do over match1;
    if match1=3 and r4wks00=>0 and r3wks001=>0 then
          do; twks00=r3wks001+r4wks00; end;
    if match1=3 and (-4<r4wks00<0 or -4<r3wks001<0)
          then do; twks00=-3; end;
  end;

/*At this point the program repeats the above code for
match2-match9. Contact NLS User Services for details.*/

end;

/* For respondents last interviewed in Round 3, 1999 */
if r3int_y=1999 then do;
  do over match1;
    if match1=3 then do;
      if r3wks801 not in (-5,-4,0,.) then
          twks80=r3wks801;
      if r3wks811 not in (-5,-4,0,.) then
          twks81=r3wks811;
        /*and so on through*/
      if r3wks981 not in (-5,-4,0,.) then
          twks98=r3wks981;
    end;
  end;

  do over match1;
    if match1=3 and r4wks99=>0 and r3wks991=>0 then
      do; twks99=r3wks991+r4wks99; end;
    if match1=3 and (-4<r4wks99<0 or -4<r3wks991<0)
      then do; twks99=-3; end;
  end;

/*At this point the program repeats the above code for
match2-match9. Contact NLS User Services for details.*/

end;

/* For respondents last interviewed in Round 2, 1998 */
if (r2int_y=1998) then do;
  do over match1;
    if match1=2 then do;
      if r2wks801 not in (-5,-4,0,.) then
          twks80=r2wks801;
      if r2wks811 not in (-5,-4,0,.) then
          twks81=r2wks811;
        /*and so on through*/
      if r2wks971 not in (-5,-4,0,.) then
          twks97=r2wks971;
    end;
  end;

  do over match1;
    if match1=2 and r4wks98=>0 and r2wks981=>0 then
      do; twks98=r2wks981+r4wks98; end;
    if match1=2 and (-4<r4wks98<0 or -4<r2wks981<0)
      then do; twks98=-3; end;
  end;

/*At this point the program repeats the above code for
match2-match9. Contact NLS User Services for details.*/

end;

/* For respondents last interviewed in Round 2, 1999 */
if r2int_y=1999 then do;
  do over match1;
    if match1=2 then do;
      if r2wks801 not in (-5,-4,0,.) then
          twks80=r2wks801;
      if r2wks811 not in (-5,-4,0,.) then
          twks81=r2wks811;
        /*and so on through*/
      if r2wks981 not in (-5,-4,0,.) then
          twks98=r2wks981;
    end;
  end;

  do over match1;
    if match1=2 and r4wks99=>0 and r2wks991=>0 then
      do; twks99=r2wks991+r4wks99; end;
    if match1=2 and (-4<r4wks99<0 or -4<r2wks991<0)
      then do; twks99=-3; end;
  end;

/*At this point the program repeats the above code for
match2-match9. Contact NLS User Services for details.*/

end;

/* For respondents last interviewed in Round 1, 1997 */
if r1int_y=1997 and intwk>0 then do;
```

```
  do over match1;
    if match1=1 then do;
      if r1wks801 not in (-5,-4,0,.) then
          twks80=r1wks801;
      if r1wks811 not in (-5,-4,0,.) then
          twks81=r1wks811;
       /*and so on through*/
      if r1wks961 not in (-5,-4,0,.) then
          twks96=r1wks961;
    end;
  end;

  do over match1;
    if match1=1 and r4wks97=>0 and r1wks971=>0 then
      do; twks97=r1wks971+r4wks97; end;
    if match1=1 and (-4<r4wks97<0 or -4<r1wks971<0)
      then do; twks97=-3; end;
  end;

/*At this point the program repeats the above code for
match2-match9. Contact NLS User Services for details.*/

end;

/* For respondents last interviewed in Round 1, 1998 */
if r1int_y=1998 and intwk>0 then do;
  do over match1;
    if match1=1 then do;
      if r1wks801 not in (-5,-4,0,.) then
          twks80=r1wks801;
      if r1wks811 not in (-5,-4,0,.) then
          twks81=r1wks811;
       /*and so on through*/
      if r1wks971 not in (-5,-4,0,.) then
          twks97=r1wks971;
    end;
  end;

  do over match1;
    if match1=1 and r4wks98=>0 and r1wks981=>0 then
      do; twks98=r1wks981+r4wks98; end;
    if match1=1 and (-4<r4wks98<0 or -4<r1wks981<0)
      then do; twks98=-3; end;
  end;

/*At this point the program repeats the above code for
match2-match9. Contact NLS User Services for details.*/

end;

/*flag problem cases where total weeks count exceeds 52*/
flag97=0; flag98=0; flag99=0;
do i=1 to 9;
  if twks97>52 then flag97=1;
  if twks98>52 then flag98=1;
  if twks99>52 then flag99=1;
  if twks00>53 then flag00=1;
end;

endsas;
```

TOTAL TENURE AT JOB #X AS OF THE SURVEY DATE

Variables Created: CV_WKSWK_JOB_DLI.01 – CV_WKSWK_JOB_DLI.09

Programs Used

This program uses **emp_begin.sas** as input (see the first page of this appendix for details).

This program creates a variable for each job calculating the total length of job tenure in weeks (excluding within-job gaps) since the respondent's 14th birthday. A variable is created for each potential job even if the respondent has no data for that job, with the default value set to zero (0). The most jobs held by any respondent as of round 2 was nine, so variables are created for nine jobs for each respondent.

```
/* Section 1:  Create a variable for each potential job (9)
relating the total length of job tenure in weeks excluding
within-job gaps.*/

array starw (i) starw1-starw9;
array job1wks (i) wk1_1-wk1_1148;
array job2wks (i) wk2_1-wk2_1148;
   /*and so on through*/
array job9wks (i) wk9_1-wk9_1148;
array starfl (i) starfl_1-starfl_9;
array stopfl (i) stopfl_1-stopfl_9;

/* Calculate cumulative weeks on individual jobs for all
years */
r4ten1=0; r4ten2=0; r4ten3=0; r4ten4=0; r4ten5=0;
r4ten6=0; r4ten7=0; r4ten8=0; r4ten9=0;

do i=1 to 1148;
  if job1wks=1 then do; r4ten1=r4ten1+1; end;
  if job2wks=1 then do; r4ten2=r4ten2+1; end;
   /*and so on through*/
  if job9wks=1 then do; r4ten9=r4ten9+1; end;
end;

flag=0;

do i=1 to 1148;
  if job1wks=-3 then do; r4ten1=-3; flag=1; end;
  if job2wks=-3 then do; r4ten2=-3; flag=2; end;
   /*and so on through*/
  if job9wks=-3 then do; r4ten9=-3; flag=9; end;
end;

do i=1 to 1096;
  if starfl_1=1 then do; r3ten1=-3; end;
  if starfl_2=1 then do; r3ten2=-3; end;
   /*and so on through*/
  if starfl_9=1 then do; r3ten9=-3; end;
  if stopfl_1=1 then do; r3ten1=-3; end;
  if stopfl_2=1 then do; r3ten2=-3; end;
   /*and so on through*/
  if stopfl_9=1 then do; r3ten9=-3; end;
end;

if e200=-5 then do;
  r4ten1=-5; r4ten2=-5; r4ten3=-5; r4ten4=-5; r4ten5=-5;
      r4ten6=-5; r4ten7=-5; r4ten8=-5; r4ten9=-5;
end;

array r3ten (i) r3ten1-r3ten9;
array totten (i) totten1-totten9;
do over totten; r3ten=totten; end;

/* 2-26-01 Begin by renaming the infile variables */
array r1ten (i) r1ten1-r1ten9; /* New R1 c.v. name */
array r2ten (i) r2ten1-r2ten9; /* New R2 c.v. name */
array r3ten (i) r3ten1-r3ten9; /* New R3 c.v. name */
array r4ten (i) r4ten1-r4ten9; /* New R4 c.v. name */
array totten (i) totten1-totten9; /* Total c.v. name */

do i=1 to 9; totten=0; end;

array r1uid (i) r1uid1-r1uid7;
array r2uid (i) r2uid1-r2uid9;
array r3uid (i) r3uid1-r3uid9;
array r4uid (i) uid1-uid9;

array match1 (i) match11-match19;
array match2 (i) match21-match29;
  /*and so on through*/
array match9 (i) match91-match99;

do i=1 to 9;
  match1=0; match2=0; match3=0; match4=0; match5=0;
      match6=0; match7=0; match8=0; match9=0;
end;

/* Define match14 as the dummy variable that equals one
when the first job on the dli UID roster and the fourth
job on the Round4 roster have the same UID. */

/* If any UID from the first position in a previous round
matches with any UID in Round 4.  Need to make sure
we get the most recent round the job was reported in. */
do over r4uid;
  if r3uid1>0 and r4uid>0 and r3int>0 and intwk>0 then
      do;
    if r3uid1=r4uid then do; match1=3; end;
```

```
  end;
  if r2uid1>0 and r4uid>0 and match1=0 and intwk>0
      then do;
    if r2uid1=r4uid then do; match1=2; end;
  end;
  if r1uid1>0 and r4uid>0 and match1=0 and intwk>0
      then do;
    if r1uid1=r4uid then do; match1=1; end;
  end;
end;

/* If any UID from the second position in a previous
      Round matches with any UID in Round 4 */
do over r4uid;
  if r3uid2>0 and r4uid>0 and r3int>0 and intwk>0 then
      do;
    if r3uid2=r4uid then do; match2=3; end;
  end;
  if r2uid2>0 and r4uid>0 and match2=0 and intwk>0
      then do;
    if r2uid2=r4uid then do; match2=2; end;
  end;
  if r1uid2>0 and r4uid>0 and match2=0 and intwk>0
      then do;
    if r1uid2=r4uid then do; match2=1; end;
  end;
end;

/*repeat the above code for UIDs in positions 3-9 in
      previous rounds*/

r4match_1=max(match11,match21,match31,match41,
      match51,match61,match71,match81,match91);
r4match_2=max(match12,match22,match32,match42,
      match52,match62,match72,match82,match92);
   /*and so on through*/
r4match_9=max(match19,match29,match39,match49,
      match59,match69,match79,match89,match99);

if match11<r4match_1 then match11=0;
if match21<r4match_1 then match21=0;
   /*and so on through*/
if match91<r4match_1 then match91=0;

if match12<r4match_2 then match12=0;
if match22<r4match_2 then match22=0;
   /*and so on through*/
if match92<r4match_2 then match92=0;

/*repeat the above code for jobs 2-9*/

/* There can only be 9 possible jobs for the respondent,
since we are not counting jobs that were only worked in
previous rounds. */
do over totten; totten=r4ten; end;

/* Now reassign total tenure if there is a job was worked in
2+ rounds. Using the match variable, a job worked in 2+
round 4 & the dli round will update the total tenure
variable. Need to add positivity condition on the tenure
values since some of the values equal -3. */

do over match1;
  if match1=3 and r4ten=>0 and r3ten1=>0 then do;
      totten=r3ten1+r4ten; end;
  if match1=3 and -4<r3ten1<0 then do; totten=-3; end;
  if match1=2 and r4ten=>0 and r2ten1=>0 then do;
      totten=r2ten1+r4ten; end;
  if match1=2 and -4<r2ten1<0 then do; totten=-3; end;
  if match1=1 and r4ten=>0 and r1ten1=>0 then do;
      totten=r1ten1+r4ten; end;
  if match1=1 and -4<r1ten1<0 then do; totten=-3; end;
end;

/*repeat the above code for match2-match9*/

endsas;
```

TOTAL HOURS WORKED IN 19XX/20XX

Variables Created: CV_HOURS_WK_YR_ALL.80 – CV_HOURS_WK_YR_ALL.01
CV_HOURS_WK_YR_ET.80 – CV_HOURS_WK_YR_ET.01
CV_HOURS_WK_YR_SE.98 – CV_HOURS_WK_YR_SE.01

Programs Used

This program uses **emp_begin.sas** as input (see the first page of this appendix for details).

This program calculates the number of hours worked by the respondent at all jobs in each calendar year. Three sets of variables are created: one for all jobs, one for employee-type jobs, and one for self-employed jobs for respondents age 18 and older in round 4. **See the introduction to this appendix for important information about the universes for each of these variables.** A variable is created for each respondent even if the respondent has worked no jobs in a given year with the default value set to zero (0). Note that when both "starting hours" and "current hours" are reported, the latter are used to construct these measures.

```
*** Organize hours for each job;
array starw (k) starw1-starw9;
array stopw (k) stopw1-stopw9;

/* shrs1   - starting hours    ehrs1   - ending hours   */
array shrs (k) shrs1-shrs9;
array ehrs (k) ehrs1-ehrs9;
array hours (k) hours1-hours9;
array hrck (k) hrck1-hrck9;

/* To the right of the array statements are the true ranges of the pulled variables. Values of variables outside the ranges
will be represented by dots. (this was done for simplicity for the programmer) */
array e23901 (k) e239011-e239019;      /* 1-9 */        array e24501 (k) e245011-e245019;      /* 1-6,9 */
array e34402 (k) e344021-e344029;      /* 1-5 */        array e34403 (k) e344031-e344039;      /* 1-3 */
array e34428 (k) e344281-e344289;      /* 1-5 */        array e37904 (k) e379041-e379049;      /* 1-5,7 */
array e38000f (k) e38000f1-e38000f9;   /* 1-5,7 */      array e38103 (k) e381031-e381039;      /* 1-3,5 */
array e38105 (k) e381051-e381059;      /* 1-3 */        array e59901 (k) e599011-e599019;      /* 1-7 */
array e88000 (k) e880001-e880009;      /* 1-7 */        array e88501 (k) e885011-e885019;      /* 1-5 */
array e98402 (k) e984021-e984209;      /* 1-6 */        array e98403 (k) e984031-e984039;      /* 1-3 */
array e98429 (k) e984291-e984299;      /* 1-3 */        array e100231 (k) e1002311-e1002319;   /* 1,2 */
array e100232 (k) e1002321-e1002329;   /* 1,2 */        array e22610 (k) e226101-e226109;      /* 1,2 */
array e22611 (k) e226111-e226119;      /* 1,2 */        array starfl (k) starfl_1-starfl_9;
array stopfl (k) stopfl_1-stopfl_9;

/* Define the number of hours per week worked at the start date and the end date of a job. Question E59901 decides
        whether the job was listed on the Round 1 roster, for jobs under 13 weeks. Question E88000 also decides
        whether the job was listed on the Round 1 roster, but only for jobs that last longer then 13 weeks. The code
        below defines the starting hours and ending hours per week at those jobs, including overtime. */
do k=1 to 9;
/* Define starting hours for jobs shorter than 13 weeks */
        if e23901>-4 then do; shrs=e23901; end;
        if e23901>-4 and e24501>-4 then do; shrs=e23901+e24501; end;
        if e34402>-4 then do; shrs=e34402;end;
        if e34403>-4 then do; shrs=e34403; end;
        if e34403>-4 and e34428>-4 then do; shrs=e34403+e34428; end;
/* Define ending hours for jobs less than 13 weeks long */  if e37904>-4 then do; ehrs=e37904; end;
/* Define ending hours for jobs longer than 13 weeks */
        if e38000f>-4 then do; ehrs=e38000f; end;
        if e38103>-4 then do; ehrs=e38103;end;
        if e38103>-4 and e38105>-4 then do; ehrs=e38103+e38105; end;
```

```
        if e59901>-4 then do; shrs=e59901; end; /* These are jobs from dli that are less than 13 weeks long */
        if e88000>-4 then do; shrs=e88000; end; /* These are jobs from dli that are longer than 13 weeks */
        if e88501>-4 and e88000>-4 then do; shrs=e88501+e88000; end; /* E88501 is an overtime question. */
        if e98402>-4 then do; shrs=e98402; end;
        if e98429>-4 then do; shrs=e98429; end;
        if e98429>-4 and e98403>-4 then do; shrs=e98403+e98429; end;
        if e22610>-4 then do; shrs=e22610; end;
        if e22611>-4 then do; shrs=e22611; end;
        if e22610>-4 and e22611>-4 then do; shrs=e22610+e22611; end;
        if e100231>-4 then do; shrs=e100231; end;
        if e100231>-4 then do; shrs=e100232; end;
        if e100231>-4 and e100232>-4 then do; shrs=e100231+e100232; end;
end;

/* The code below decides which hours per week total (starting or ending) will be used in determining hours worked per
        year.  We prefer using ending hours as the measure, hours per week starts with a default number (-16), is
        updating if starting hours are available, and is updated once more if ending hours are available. */
* Set default hours to zero (-16);
do over hours; hours=0; hrck=0; end;

* Take starting hours if reported (eliminates -4's);
do over shrs;
  if shrs=>0 then do; hours=shrs; end;
    /* hrck marks when invalid answers are given to the hrs/wk questions */
  if -4<shrs<0 then do; hrck=shrs; end;
end;

* Write over if end hours reported;
do over ehrs;
  if ehrs=>0 then do; hours=ehrs; end;
  if -4<ehrs<0 then do; hrck=ehrs; end;
end;

* Check for respondents who report valid starting hours and invalid ending hours...(11-29-01) There are 10 cases, all of
        which have ehrs=-2. Since the respondent reports -2's, the c.v. will be -2 as well.;
do over ehrs;
  if shrs>0 and -4<ehrs<0 then do; hrprob=1; end;
end;

/* This checks the people that report too many hours per week. */
do over hours; if hours=>60 then noway=1; end;

/* This fixes a person who has the job start & stop data at loop 4 and the hours data in loop three. Other than the hours
        data, there is no other employment data in loop 3. */
if id=1024861 then do; hours3=.; hours4=14; end;

array job1wks (i) wk1_1-wk1_1148;     array job2wks (i) wk2_1-wk2_1148;     array job3wks (i) wk3_1-wk3_1148;
array job4wks (i) wk4_1-wk4_1148;     array job5wks (i) wk5_1-wk5_1148;     array job6wks (i) wk6_1-wk6_1148;
array job7wks (i) wk7_1-wk7_1148;     array job8wks (i) wk8_1-wk8_1148;     array job9wks (i) wk9_1-wk9_1148;

/** Calculate cumulative weeks on individual jobs for each year **/
/* 1980 */
array wks80 (i) wks801-wks809;     array swks80 (i) swks801-swks809;   array ewks80 (i) ewks801-ewks809;
array ah80 (i) ah801-ah809;        array sah80 (i) sah801-sah809;      array eah80 (i) eah801-eah809;

do i=1 to 9; wks80=0; swks80=0; ewks80=0; ah80=0; sah80=0; eah80=0; end;
```

```
/* Update hours counter when respondent reports employment in weeks 1 to 52. */
do i=1 to 52;
    if job1wks=1 then do; wks801=wks801+1; end;        if job2wks=1 then do; wks802=wks802+1; end;
    if job3wks=1 then do; wks803=wks803+1; end;        if job4wks=1 then do; wks804=wks804+1; end;
    if job5wks=1 then do; wks805=wks805+1; end;        if job6wks=1 then do; wks806=wks806+1; end;
    if job7wks=1 then do; wks807=wks807+1; end;        if job8wks=1 then do; wks808=wks808+1; end;
    if job9wks=1 then do; wks809=wks809+1; end;

    if job1wks=1 and self1=1 then do; swks801=swks801+1; end;        /* Self-employed jobs */
    if job2wks=1 and self2=1 then do; swks802=swks802+1; end;
      /*and so on through*/
    if job9wks=1 and self9=1 then do; swks809=swks809+1; end;

    if job1wks=1 and self1=0 then do; ewks801=ewks801+1; end;        /* Employee-type jobs */
    if job2wks=1 and self2=0 then do; ewks802=ewks802+1; end;
      /*and so on through*/
    if job9wks=1 and self9=0 then do; ewks809=ewks809+1; end;
end;

do i=1 to 52;
  if hours1>0 and job1wks ne -3 then ah801=hours1*wks801;
  if hours2>0 and job2wks ne -3 then ah802=hours2*wks802;
      /*and so on through*/
  if hours9>0 and job9wks ne -3 then ah809=hours9*wks809;

  if hours1>0 and job1wks ne -3 and self1=1 then sah801=hours1*swks801;    /* Self-employed jobs */
  if hours2>0 and job2wks ne -3 and self2=1 then sah802=hours2*swks802;
      /*and so on through*/
  if hours9>0 and job9wks ne -3 and self9=1 then sah809=hours9*swks809;

  if hours1>0 and job1wks ne -3 and self1=0 then eah801=hours1*ewks801;    /* Employee-type jobs */
  if hours2>0 and job2wks ne -3 and self2=0 then eah802=hours2*ewks802;
      /*and so on through*/
  if hours9>0 and job9wks ne -3 and self9=0 then eah809=hours9*ewks809;
end;

do i=1 to 52;
  if job1wks=-3 or 0<hours1<-4 then ah801=-3;                /* Define negative values for each job separately. */
  if (job1wks=-3 or 0<hours1<-4) and self1=1 then sah801=-3;
  if (job1wks=-3 or 0<hours1<-4) and self1=0 then eah801=-3;
  if job2wks=-3 or 0<hours2<-4 then ah802=-3;
  if (job2wks=-3 or 0<hours2<-4) and self2=1 then sah802=-3;
  if (job2wks=-3 or 0<hours2<-4) and self2=0 then eah802=-3;
      /*and so on through*/
  if job9wks=-3 or 0<hours9<-4 then ah809=-3;
  if (job9wks=-3 or 0<hours9<-4) and self9=1 then sah809=-3;
  if (job9wks=-3 or 0<hours9<-4) and self9=0 then eah809=-3;
end;

/* Sum up the valid hours of each job */
if ah801=>0 and ah802=>0 and ah803=>0 and ah804=>0 and ah805=>0 and ah806=>0 and ah807=>0 and ah808=>0
        and ah809=>0 then do;
  r4hrs80=ah801+ah802+ah803+ah804+ah805+ah806+ah807+ah808+ah809;
end;
if sah801=>0 and sah802=>0 and sah803=>0 and sah804=>0 and sah805=>0 and sah806=>0 and sah807=>0 and
        sah808=>0 and sah809=>0 then do;
  r4shrs80=sah801+sah802+sah803+sah804+sah805+sah806+sah807+sah808+sah809;
```

```
end;
if eah801=>0 and eah802=>0 and eah803=>0 and eah804=>0 and eah805=>0 and eah806=>0 and eah807=>0 and
        eah808=>0 and eah809=>0 then do;
  r4ehrs80=eah801+eah802+eah803+eah804+eah805+eah806+eah807+eah808+eah809;
end;

/* Place in invalid skips */
if ah801=-3 or ah802=-3 or ah803=-3 or ah804=-3 or ah805=-3 or ah806=-3 or ah807=-3 or ah808=-3 or ah809=-3 then
        do; r4hrs80=-3; end;
if sah801=-3 or sah802=-3 or sah803=-3 or sah804=-3 or sah805=-3 or sah806=-3 or sah807=-3 or sah808=-3 or
        sah809=-3 then do; r4shrs80=-3; end;
if eah801=-3 or eah802=-3 or eah803=-3 or eah804=-3 or eah805=-3 or eah806=-3 or eah807=-3 or eah808=-3 or
        eah809=-3 then do; r4ehrs80=-3; end;
```

/* At this point the program repeats the above 1980 code for each year through 2001. This code is omitted here due to space considerations. Users who need to see the entire program should contact NLS User Services. The week numbers and variables created for each year are as follows:

Year/ Weeks	Variables Created		Year/ Weeks	Variables Created	
1981 53-104	wks811-wks819; swks811-swks819; ewks811-ewks819;	ah811-ah819; sah811-sah819; eah811-eah819;	1992 627-678	wks921-wks929; swks921-swks929; ewks921-ewks929;	ah921-ah929; sah921-sah929; eah921-eah929;
1982 105-156	wks821-wks829; swks821-swks829; ewks821-ewks829;	ah821-ah829; sah821-sah829; eah821-eah829;	1993 679-730	wks931-wks939; swks931-swks939; ewks931-ewks939;	ah931-ah939; sah931-sah939; eah931-eah939;
1983 157-209	wks831-wks839; swks831-swks839; ewks831-ewks839;	ah831-ah839; sah831-sah839; eah831-eah839;	1994 731-783	wks941-wks949; swks941-swks949; ewks941-ewks949;	ah941-ah949; sah941-sah949; eah941-eah949;
1984 210-261	wks841-wks849; swks841-swks849; ewks841-ewks849;	ah841-ah849; sah841-sah849; eah841-eah849;	1995 784-835	wks951-wks959; swks951-swks959; ewks951-ewks959;	ah951-ah959; sah951-sah959; eah951-eah959;
1985 262-313	wks851-wks859; swks851-swks859; ewks851-ewks859;	ah851-ah859; sah851-sah859; eah851-eah859;	1996 836-887	wks961-wks969; swks961-swks969; ewks961-ewks969;	ah961-ah969; sah961-sah969; eah961-eah969;
1986 314-365	wks861-wks869; swks861-swks869; ewks861-ewks869;	ah861-ah869; sah861-sah869; eah861-eah869;	1997 888-939	wks971-wks979; swks971-swks979; ewks971-ewks979;	ah971-ah979; sah971-sah979; eah971-eah979;
1987 366-417	wks871-wks879; swks871-swks879; ewks871-ewks879;	ah871-ah879; sah871-sah879; eah871-eah879;	1998 940-991	wks981-wks989; swks981-swks989; ewks981-ewks989;	ah981-ah989; sah981-sah989; eah981-eah989;
1988 418-470	wks881-wks889; swks881-swks889; ewks881-ewks889;	ah881-ah889; sah881-sah889; eah881-eah889;	1999 992-1043	wks991-wks999; swks991-swks999; ewks991-ewks999;	ah991-ah999; sah991-sah999; eah991-eah999;
1989 471-522	wks891-wks899; swks891-swks899; ewks891-ewks899;	ah891-ah899; sah891-sah899; eah891-eah899;	2000 1044-1096	wks001-wks009; swks001-swks009; ewks001-ewks009;	ah001-ah009; sah001-sah009; eah001-eah009;
1990 523-574	wks901-wks909; swks901-swks909; ewks901-ewks909;	ah901-ah909; sah901-sah909; eah901-eah909;	2001 1097-1148	wks011-wks019; swks011-swks019; ewks011-ewks019;	ah011-ah019; sah011-sah019; eah011-eah019;
1991 575-626	wks911-wks919; swks911-swks919; ewks911-ewks919;	ah911-ah919; sah911-sah919; eah911-eah919;			

```
if e200 in (-3,-5) then do;
```

```
  r4hrs80=-5; r4hrs81=-5; r4hrs82=-5; r4hrs83=-5; r4hrs84=-5; r4hrs85=-5; r4hrs86=-5; r4hrs87=-5; r4hrs88=-5;
        r4hrs89=-5; r4hrs90=-5; r4hrs91=-5; r4hrs92=-5; r4hrs93=-5; r4hrs94=-5; r4hrs95=-5; r4hrs96=-5; r4hrs97=-5;
        r4hrs98=-5; r4hrs99=-5; r4hrs00=-5; r4hrs01=-5;
  r4shrs80=-5; r4shrs81=-5; r4shrs82=-5; r4shrs83=-5; r4shrs84=-5; r4shrs85=-5; r4shrs86=-5; r4shrs87=-5; r4shrs88=-5;
        r4shrs89=-5; r4shrs90=-5; r4shrs91=-5; r4shrs92=-5; r4shrs93=-5; r4shrs94=-5; r4shrs95=-5; r4shrs96=-5;
        r4shrs97=-5; r4shrs98=-5; r4shrs99=-5; r4shrs00=-5; r4shrs01=-5;
  r4ehrs80=-5; r4ehrs81=-5; r4ehrs82=-5; r4ehrs83=-5; r4ehrs84=-5; r4ehrs85=-5; r4ehrs86=-5; r4ehrs87=-5; r4ehrs88=-
        5; r4ehrs89=-5; r4ehrs90=-5; r4ehrs91=-5; r4ehrs92=-5; r4ehrs93=-5; r4ehrs94=-5; r4ehrs95=-5; r4ehrs96=-5;
        r4ehrs97=-5; r4ehrs98=-5; r4ehrs99=-5; r4ehrs00=-5; r4ehrs01=-5;
end;

/* Fill in -3's for cases where start/stop date is unknown/refused. */
do k=1 to 9;

/* start date invalid */
if starfl=1 and tothrs80>0 and stopw>1 then do; tothrs80=-3; end;
if starfl=1 and tothrs81>0 and stopw>52 then do; tothrs81=-3; end;
if starfl=1 and tothrs82>0 and stopw>104 then do; tothrs82=-3; end;
if starfl=1 and tothrs83>0 and stopw>156 then do; tothrs83=-3; end;
if starfl=1 and tothrs84>0 and stopw>209 then do; tothrs84=-3; end;
if starfl=1 and tothrs85>0 and stopw>261 then do; tothrs85=-3; end;
if starfl=1 and tothrs86>0 and stopw>313 then do; tothrs86=-3; end;
if starfl=1 and tothrs87>0 and stopw>365 then do; tothrs87=-3; end;
if starfl=1 and tothrs88>0 and stopw>417 then do; tothrs88=-3; end;
if starfl=1 and tothrs89>0 and stopw>470 then do; tothrs89=-3; end;
if starfl=1 and tothrs90>0 and stopw>522 then do; tothrs90=-3; end;
if starfl=1 and tothrs91>0 and stopw>574 then do; tothrs91=-3; end;
if starfl=1 and tothrs92>0 and stopw>626 then do; tothrs92=-3; end;
if starfl=1 and tothrs93>0 and stopw>678 then do; tothrs93=-3; end;
if starfl=1 and tothrs94>0 and stopw>730 then do; tothrs94=-3; end;
if starfl=1 and tothrs95>0 and stopw>783 then do; tothrs95=-3; end;
if starfl=1 and tothrs96>0 and stopw>835 then do; tothrs96=-3; end;
if starfl=1 and tothrs97>0 and stopw>887 then do; tothrs97=-3; end;
if starfl=1 and tothrs98>0 and stopw>939 then do; tothrs98=-3; end;
if starfl=1 and tothrs99>0 and stopw>990 then do; tothrs99=-3; end;
if starfl=1 and tothrs00>0 and stopw>1043 then do; tothrs00=-3; end;
if starfl=1 and tothrs01>0 and stopw>1096 then do; tothrs01=-3; end;

/* stop date invalid */
if stopfl=1 and tothrs80>0 and starw<53 then do; tothrs80=-3; end;
if stopfl=1 and tothrs81>0 and starw<105 then do; tothrs81=-3; end;
if stopfl=1 and tothrs82>0 and starw<157 then do; tothrs82=-3; end;
if stopfl=1 and tothrs83>0 and starw<210 then do; tothrs83=-3; end;
if stopfl=1 and tothrs84>0 and starw<262 then do; tothrs84=-3; end;
if stopfl=1 and tothrs85>0 and starw<314 then do; tothrs85=-3; end;
if stopfl=1 and tothrs86>0 and starw<366 then do; tothrs86=-3; end;
if stopfl=1 and tothrs87>0 and starw<418 then do; tothrs87=-3; end;
if stopfl=1 and tothrs88>0 and starw<471 then do; tothrs88=-3; end;
if stopfl=1 and tothrs89>0 and starw<523 then do; tothrs89=-3; end;
if stopfl=1 and tothrs90>0 and starw<575 then do; tothrs90=-3; end;
if stopfl=1 and tothrs91>0 and starw<627 then do; tothrs91=-3; end;
if stopfl=1 and tothrs92>0 and starw<679 then do; tothrs92=-3; end;
if stopfl=1 and tothrs93>0 and starw<731 then do; tothrs93=-3; end;
if stopfl=1 and tothrs94>0 and starw<784 then do; tothrs94=-3; end;
if stopfl=1 and tothrs95>0 and starw<836 then do; tothrs95=-3; end;
if stopfl=1 and tothrs96>0 and starw<888 then do; tothrs96=-3; end;
if stopfl=1 and tothrs97>0 and starw<940 then do; tothrs97=-3; end;
```

```
if stopfl=1 and tothrs98>0 and starw<991 then do; tothrs98=-3; end;
if stopfl=1 and tothrs99>0 and starw<1044 then do; tothrs99=-3; end;
if stopfl=1 and tothrs00>0 and starw<1096 then do; tothrs00=-3; end;
if stopfl=1 and tothrs01>0 and starw<1149 then do; tothrs01=-3; end;
end;

/*****this section merges the round 4 data with the data from previous rounds*****/
array thrs (i) thrs80-thrs99 thrs00 thrs01;          array tshrs (i) tshrs80-tshrs99 tshrs00 tshrs01;
array tehrs (i) tehrs80-tehrs99 tehrs00 tehrs01;     array r1hrs (i) r1hrs80-r1hrs99 r1hrs00 r1hrs01;
array r2hrs (i) r2hrs80-r2hrs99 r2hrs00 r2hrs01;     array r3hrs (i) r3hrs80-r3hrs99 r3hrs00 r3hrs01;
array r4hrs (i) r4hrs80-r4hrs99 r4hrs00 r4hrs01;     array r4shrs (i) r4shrs80-r4shrs99 r4shrs00 r4shrs01;
array r4ehrs (i) r4ehrs80-r4ehrs99 r4ehrs00 r4ehrs01;

/* Begin by splitting up periods where Round 4 and dli Round exclusively collect hours worked information. Any hours
        worked information collected before the dli Round interview year should be independent of information
        collected in Round 4. For the new Round 4 c.v.'s, we assume that all dli jobs are employee-type jobs. Thus, dli
        information will be added to the "all jobs" and "employee jobs" c.v.'s but not "self-employed jobs" c.v.'s. */
do over r4hrs;
  thrs=0; tshrs=0; tehrs=0;
  if r4hrs>0 then do; thrs=r4hrs; tehrs=r4ehrs; end;
  if r4shrs>0 then do; tshrs=r4shrs;end;
  if r3hrs>0 and dliwk=r3int then do; thrs=r3hrs; tehrs=r3hrs; end;
  if r2hrs>0 and dliwk=r2int then do; thrs=r2hrs; tehrs=r2hrs; end;
  if r1hrs>0 and dliwk=r1int then do; thrs=r1hrs; tehrs=r1hrs; end;
  if r3hrs>0 and r4hrs>0 and dliwk=r3int then do; thrs=r3hrs+r4hrs; end;
  if r3hrs>0 and r4ehrs>0 and dliwk=r3int then do; tehrs=r3hrs+r4ehrs; end;
  if r2hrs>0 and r4hrs>0 and dliwk=r2int then do; thrs=r2hrs+r4hrs; end;
  if r2hrs>0 and r4ehrs>0 and dliwk=r2int then do; tehrs=r2hrs+r4ehrs; end;
  if r1hrs>0 and r4hrs>0 and dliwk=r1int then do; thrs=r1hrs+r4hrs; end;
  if r1hrs>0 and r4ehrs>0 and dliwk=r1int then do; tehrs=r1hrs+r4ehrs; end;
end;

do over r4hrs;
  if -4<r4hrs<0 then do; thrs=r4hrs; end;
  if -4<r4shrs<0 then do; tshrs=r4shrs; end;
  if -4<r4ehrs<0 then do; tehrs=r4ehrs; end;
  if -4<r3hrs<0 and dliwk=r3int then do; thrs=r3hrs; tehrs=r3hrs; end;
  if -4<r2hrs<0 and dliwk=r2int then do; thrs=r2hrs; tehrs=r2hrs; end;
  if -4<r1hrs<0 and dliwk=r1int then do; thrs=r1hrs; tehrs=r1hrs; end;
end;

do over thrs;
  if e200=-5 then do; thrs=-5; tehrs=-5; tshrs=-5; end;
end;

endsas;
```

TOTAL HOURS WORKED AGE 14-19, HOURS WORKED AT ALL JOBS SINCE AGE 20, AND HOURS WORKED AT EMPLOYEE-TYPE JOBS SINCE AGE 20

Variables Created: CV_HOURS_WK_TEEN
CV_HOURS_WK_ADULT_ALL
CV_HOURS_WK_ADULT_ET

Programs Used

This program uses **emp_begin.sas** as input (see the first page of this appendix for details).

This program calculates the number of hours worked by the respondent at all employee-type jobs between the ages of 14 and 19. It then creates two variables for the number of hours worked by the respondent since age 20: one for all jobs and one for employee-type jobs. **See the introduction to this appendix for important information about the universes for each of these variables.** A variable is created for each respondent even if the respondent has worked no jobs in a given year with the default value set to zero (0). Note that when both "starting hours" and "current hours" are reported, the latter are used to construct these measures.

```
*** Organize hours for each job;
array starw (i) starw1-starw9;
array stopw (i) stopw1-stopw9;
array starfl (i) starfl_1-starfl_9;
array stopfl (i) stopfl_1-stopfl_9;

/* shrs1   - starting hours    ehrs1   - ending hours   */

array shrs (i) shrs1-shrs9;
array ehrs (i) ehrs1-ehrs9;
array hours (i) hours1-hours9;
array hrck (i) hrck1-hrck9;

/* To the right of the array statements are the true ranges
of the pulled variables. Values of variables outside the
ranges will be represented by dots. (this was done for
simplicity for the programmer) */
array e22610 (i) e226101-e226109;
array e22611 (i) e226111-e226119;
array e100231 (i) e1002311-e1002319;
array e100232 (i) e1002321-e1002329;
array e23901 (i) e239011-e239019;          /* 1-9 */
array e24501 (i) e245011-e245019;          /* 1-6,9 */
array e34402 (i) e344021-e344029;          /* 1-5 */
array e34403 (i) e344031-e344039;          /* 1-3 */
array e34428 (i) e344281-e344289;          /* 1-5 */
array e37904 (i) e379041-e379049;          /* 1-5,7 */
array e38000f (i) e38000f1-e38000f9;       /* 1-5,7 */
array e38103 (i) e381031-e381039;          /* 1-3,5 */
array e38105 (i) e381051-e381059;          /* 1-3 */
array e59901 (i) e599011-e599019;          /* 1-7 */
array e88000 (i) e880001-e880009;          /* 1-7 */
array e88501 (i) e885011-e885019;          /* 1-5 */
array e98402 (i) e984021-e984209;          /* 1-6 */
array e98403 (i) e984031-e984039;          /* 1-3 */
array e98429 (i) e984291-e984299;          /* 1-3 */
array e100256 (i) e1002561-e1002569;       /* 1,2 */
```

```
/* Define the number of hours per week worked at the
start date and the end date of a job. Question E59901
decides whether the job was listed on the Round 1 roster,
for jobs under 13 weeks. Question E88000 also decides
whether the job was listed on the Round 1 roster, but only
for jobs that last longer then 13 weeks. The code below
defines the starting hours and ending hours per week at
those jobs, including overtime. */

do i=1 to 9;
/* Define starting hours for jobs shorter than 13 weeks */
  if e23901>-4 then do; shrs=e23901; end;
  if e23901>-4 and e24501>-4 then do;
      shrs=e23901+e24501; end;
  if e34402>-4 then do; shrs=e34402; end;
  if e34403>-4 then do; shrs=e34403; end;
  if e34403>-4 and e34428>-4 then do;
      shrs=e34403+e34428; end;
/* Define ending hours for jobs less than 13 weeks long */
  if e37904>-4 then do; ehrs=e37904; end;
/* Define ending hours for jobs longer than 13 weeks */
  if e38000f>-4 then do; ehrs=e38000f; end;
  if e38103>-4 then do; ehrs=e38103; end;
  if e38103>-4 and e38105>-4 then do;
      ehrs=e38103+e38105; end;
/*These are jobs from dli less than 13 weeks long*/
  if e59901>-4 then do; shrs=e59901; end;
/* These are jobs from dli that are longer than 13 weeks */
  if e88000>-4 then do; shrs=e88000; end;
/* E88501 is an overtime question. */
  if e88501>-4 and e88000>-4 then do;
      shrs=e88501+e88000; end;
  if e98402>-4 then do; shrs=e98402; end;
  if e98429>-4 then do; shrs=e98429; end;
  if e98429>-4 and e98403>-4 then do;
      shrs=e98403+e98429; end;
  if e22610>-4 then do; shrs=e22610; end;
  if e22611>-4 then do; shrs=e22611; end;
```

```
if e22610>-4 and e22611>-4 then do;
        shrs=e22610+e22611; end;
if e100231>-4 then do; shrs=e100231; end;
if e100231>-4 then do; shrs=e100232; end;
if e100231>-4 and e100232>-4 then do;
        shrs=e100231+e100232; end;
end;

/* The code below decides which hours per week total
(starting or ending) will be used in determining hours
worked per year. We prefer using ending hours as the
measure, hours per week starts with a default number (-
16), is updating if starting hours are available, and is
updated once more if ending hours are available. */

* Set default hours to zero (-16);
do over hours; hours=0; hrck=0; end;

* Take starting hours if reported (eliminates -4's);
do over shrs;
   if shrs=>0 then do; hours=shrs; end;
/* hrck marks invalid answers for the hrs/wk questions */
   if -4<shrs<0 then do; hrck=shrs; end;
end;

* Write over if end hours reported;
do over ehrs;
   if ehrs=>0 then do; hours=ehrs; end;
   if -4<ehrs<0 then do; hrck=ehrs; end;
end;

array job1wks (i) wk1_1-wk1_1148;
array job2wks (i) wk2_1-wk2_1148;
array job3wks (i) wk3_1-wk3_1148;
array job4wks (i) wk4_1-wk4_1148;
array job5wks (i) wk5_1-wk5_1148;
array job6wks (i) wk6_1-wk6_1148;
array job7wks (i) wk7_1-wk7_1148;
array job8wks (i) wk8_1-wk8_1148;
array job9wks (i) wk9_1-wk9_1148;

/** Calculate cumulative weeks on employee-type jobs for
each year ** /

/* Initialize created variables */
r4hrs14=0; r4hrs20=0; r4ehrs20=0;

/* Read "wks145" as the number of weeks worked
between the ages of 14 and 19, for job 5 only. Read
"thrs145" as the number of hours worked between the
ages of 14 and 19 for job 5 only. */
wks141=0; wks142=0; wks143=0; wks144=0; wks145=0;
wks146=0; wks147=0; wks148=0; wks149=0; thrs141=0;
thrs142=0; thrs143=0; thrs144=0; thrs145=0; thrs146=0;
thrs147=0; thrs148=0; thrs149=0;

/* Respondents who have not turned 20 by Round 4. */
if age14wk>0 and (age20wk-1)>1148 then do;
 do i=age14wk to 1148;
   if job1wks=1 and self1=0 then do; wks141=wks141+1;
        end;
   if job2wks=1 and self2=0 then do; wks142=wks142+1;
        end;
     /*and so on through*/
   if job9wks=1 and self9=0 then do; wks149=wks149+1;
        end;
 end;

 do i=age14wk to 1148;
   if job1wks ne -3 and hours1=>0 and self1=0 then
        thrs141=hours1*wks141;
   if job2wks ne -3 and hours2=>0 and self2=0 then
        thrs142=hours2*wks142;
     /*and so on through*/
   if job9wks ne -3 and hours9=>0 and self9=0 then
        thrs149=hours9*wks149;
 end;

 do i=age14wk to 1148;
   if (job1wks=-3 and self1=0) then thrs141=-3;
   if (job2wks=-3 and self2=0) then thrs142=-3;
     /*and so on through*/
   if (job9wks=-3 and self9=0) then thrs149=-3;
end;

if thrs141=>0 and thrs142=>0 and thrs143=>0 and
        thrs144=>0 and thrs145=>0 and thrs146=>0 and
        thrs147=>0 and thrs148=>0 and thrs149=>0 then
        do;
   r4hrs14=thrs141 + thrs142 + thrs143 + thrs144 +
        thrs145 + thrs146 + thrs147 + thrs148 + thrs149;
end;
if (thrs141=-3 and self1=0) or (thrs142=-3 and self2=0) or
        (thrs143=-3 and self3=0) or (thrs144=-3 and
        self4=0) or (thrs145=-3 and self5=0) or (thrs146=-3
        and self6=0) or (thrs147=-3 and self7=0) or
        (thrs148=-3 and self8=0) or (thrs149=-3 and
        self9=0) then do;
   r4hrs14=-3;
end;

/* hrck is used when invalid hrs/wk answers are given */
if wks141>0 and hrck1<0 and r4hrs14>0 and self1=0 then
        r4hrs14=-3;
if wks142>0 and hrck2<0 and r4hrs14>0 and self2=0 then
        r4hrs14=-3;
/*and so on through */
if wks149>0 and hrck9<0 and r4hrs14>0 and self9=0 then
        r4hrs14=-3;
end;

/* Respondents who have turned 20 by Round 3. */
if age14wk>0 and (age20wk-1)<=1148 then do;
 do i=age14wk to (age20wk-1);
```

```
  if job1wks=1 and self1=0 then do; wks141=wks141+1;
      end;
  if job2wks=1 and self2=0 then do; wks142=wks142+1;
      end;
    /*and so on through*/
  if job9wks=1 and self9=0 then do; wks149=wks149+1;
      end;
end;

do i=age14wk to (age20wk-1);
  if job1wks ne -3 and hours1=>0 and self1=0 then
      thrs141=hours1*wks141;
  if job2wks ne -3 and hours2=>0 and self2=0 then
      thrs142=hours2*wks142;
    /*and so on through*/
  if job9wks ne -3 and hours9=>0 and self9=0 then
      thrs149=hours9*wks149;
end;

do i=age14wk to (age20wk-1);
  if (job1wks=-3 and self1=0) then thrs141=-3;
  if (job2wks=-3 and self2=0) then thrs142=-3;
    /*and so on through*/
  if (job9wks=-3 and self9=0) then thrs149=-3;
end;

if thrs141=>0 and thrs142=>0 and thrs143=>0 and
      thrs144=>0 and thrs145=>0 and thrs146=>0 and
      thrs147=>0 and thrs148=>0 and thrs149=>0 then
      do;
  r4hrs14=thrs141 + thrs142 + thrs143 + thrs144 +
      thrs145 + thrs146 + thrs147 + thrs148 + thrs149;
end;
if (thrs141=-3 and self1=0) or (thrs142=-3 and self2=0)
      or (thrs143=-3 and self3=0) or (thrs144=-3 and
      self4=0) or (thrs145=-3 and self5=0) or (thrs146=-3
      and self6=0) or (thrs147=-3 and self7=0) or
      (thrs148=-3 and self8=0) or (thrs149=-3 and
      self9=0) then do;
  r4hrs14=-3;
end;

/* hrck is used when invalid hrs./wk answers are given */
if wks141>0 and hrck1<0 and r4hrs14>0 and self1=0
      then r4hrs14=-3;
if wks142>0 and hrck2<0 and r4hrs14>0 and self2=0
      then r4hrs14=-3;
    /*and so on through*/
if wks149>0 and hrck9<0 and r4hrs14>0 and self9=0
      then r4hrs14=-3;
end;

/* Read "wks205" as the number of weeks worked after
age 20 for job 5 only. Read "thrs205" as the number of
hours worked after age 20 for job 5 only. */
wks201=0; wks202=0; wks203=0; wks204=0; wks205=0;
      wks206=0; wks207=0; wks208=0; wks209=0;
thrs201=0; thrs202=0; thrs203=0; thrs204=0; thrs205=0;
      thrs206=0; thrs207=0; thrs208=0; thrs209=0;

/* Respondents who have turned 20 by Round 3. */
if age14wk>0 and (age20wk-1)<=1148 then do;
 do i=age20wk to 1148;
  if job1wks=1 then do; wks201=wks201+1; end;
  if job2wks=1 then do; wks202=wks202+1; end;
    /*and so on through*/
  if job9wks=1 then do; wks209=wks209+1; end;
 end;

 do i=age20wk to 1148;
  if job1wks ne -3 and hours1=>0 then
      thrs201=hours1*wks201;
  if job2wks ne -3 and hours2=>0 then
      thrs202=hours2*wks202;
    /*and so on through*/
  if job9wks ne -3 and hours9=>0 then
      thrs209=hours9*wks209;
 end;

 do i=age20wk to 1148;
  if (job1wks=-3) then thrs201=-3;
  if (job2wks=-3) then thrs202=-3;
    /*and so on through*/
  if (job9wks=-3) then thrs209=-3;
 end;

if thrs201=>0 and thrs202=>0 and thrs203=>0 and
      thrs204=>0 and thrs205=>0 and thrs206=>0 and
      thrs207=>0 and thrs208=>0 and thrs209=>0 then
      do;
  r4hrs20=thrs201 + thrs202 + thrs203 + thrs204 +
      thrs205 + thrs206 + thrs207 + thrs208 + thrs209;
end;
if thrs201=-3 or thrs202=-3 or thrs203=-3 or thrs204=-3
      or thrs205=-3 or thrs206=-3 or thrs207=-3 or
      thrs208=-3 or thrs209=-3 then do;
  r4hrs20=-3;
end;

/* hrck is used when invalid hrs./wk answers are given */
if wks201>0 & hrck1<0 & r4hrs20>0 then r4hrs20=-3;
if wks202>0 & hrck2<0 & r4hrs20>0 then r4hrs20=-3;
    /*and so on through*/
if wks209>0 & hrck9<0 & r4hrs20>0 then r4hrs20=-3;
end;

/* Read "wks205" as the number of weeks worked after
age 20 for job 5 only. Read "thrs205" as the number of
hours worked after age 20 for job 5 only. */
ewks201=0; ewks202=0; ewks203=0; ewks204=0;
      ewks205=0; ewks206=0; ewks207=0; ewks208=0;
      ewks209=0;
```

```
tehrs201=0; tehrs202=0; tehrs203=0; tehrs204=0;
        tehrs205=0; tehrs206=0; tehrs207=0; tehrs208=0;
        tehrs209=0;

/* Respondents who have turned 20 by Round 3. */
if age14wk>0 and (age20wk-1)<=1148 then do;
 do i=age20wk to 1148;
   if job1wks=1 and self1=0 then do;
        ewks201=ewks201+1; end;
   if job2wks=1 and self2=0 then do;
        ewks202=ewks202+1; end;
      /*and so on through*/
   if job9wks=1 and self9=0 then do;
        ewks209=ewks209+1; end;
 end;

 do i=age20wk to 1148;
   if job1wks ne -3 and hours1=>0 and self1=0 then
        tehrs201=hours1*ewks201;
   if job2wks ne -3 and hours2=>0 and self2=0 then
        tehrs202=hours2*ewks202;
      /*and so on through*/
   if job9wks ne -3 and hours9=>0 and self9=0 then
        tehrs209=hours9*ewks209;
 end;

 do i=age20wk to 1148;
   if (job1wks=-3 and self1=0) then tehrs201=-3;
   if (job2wks=-3 and self2=0) then tehrs202=-3;
      /*and so on through*/
   if (job9wks=-3 and self9=0) then tehrs209=-3;
 end;

if tehrs201=>0 and tehrs202=>0 and tehrs203=>0 and
        tehrs204=>0 and tehrs205=>0 and tehrs206=>0
        and tehrs207=>0 and tehrs208=>0 and
        tehrs209=>0 then do;
   r4ehrs20=tehrs201+tehrs202+tehrs203+tehrs204+
        tehrs205+tehrs206+tehrs207+tehrs208+tehrs209;
end;

if tehrs201=-3 or tehrs202=-3 or tehrs203=-3 or
        tehrs204=-3 or tehrs205=-3 or tehrs206=-3 or
        tehrs207=-3 or tehrs208=-3 or tehrs209=-3 then
        do;
   r4ehrs20=-3;
end;

/* hrck is used when invalid hrs./wk answers are given */
if ewks201>0 and hrck1<0 and r4ehrs20>0 and self1=0
        then r4ehrs20=-3;
if ewks202>0 and hrck2<0 and r4ehrs20>0 and self2=0
        then r4ehrs20=-3;
      /*and so on through*/
if ewks209>0 and hrck9<0 and r4ehrs20>0 and self9=0
        then r4ehrs20=-3;
end;

array self (i) self1-self9;

/* Remove invalid start/stop dates */
do i=1 to 9;
if age14wk<784 then do; if starfl=1 and r4hrs14>0 and
        stopw>730 then do; r4hrs14=-3; end; end;
if age14wk<836 then do; if starfl=1 and r4hrs14>0 and
        stopw>783 then do; r4hrs14=-3; end; end;
if age14wk<888 then do; if starfl=1 and r4hrs14>0 and
        stopw>835 then do; r4hrs14=-3; end; end;
if age14wk<940 then do; if starfl=1 and r4hrs14>0 and
        stopw>887 then do; r4hrs14=-3; end; end;
if age14wk<991 then do; if starfl=1 and r4hrs14>0 and
        stopw>939 then do; r4hrs14=-3; end; end;
if age14wk<1044 then do; if starfl=1 and r4hrs14>0 and
        stopw>990 then do; r4hrs14=-3; end; end;
if age14wk<1097 then do; if starfl=1 and r4hrs14>0 and
        stopw>1043 then do; r4hrs14=-3; end; end;
if age20wk<1097 then do; if starfl=1 and r4hrs20>0 and
        stopw>1043 then do; r4hrs20=-3; end; end;
if age20wk<1097 then do; if starfl=1 and self=0 and
        r4ehrs20>0 and stopw>1043 then do; r4ehrs20=-3;
        end; end;
if age14wk<1149 then do; if starfl=1 and r4hrs14>0 and
        stopw>1096 then do; r4hrs14=-3; end; end;
if age20wk<1149 then do; if starfl=1 and r4hrs20>0 and
        stopw>1096 then do; r4hrs20=-3; end; end;
if age20wk<1149 then do; if starfl=1 and self=0 and
        r4ehrs20>0 and stopw>1096 then do; r4ehrs20=-3;
        end; end;

if age14wk<784 then do; if stopfl=1 and r4hrs14>0 and
        starw<784 then do; r4hrs14=-3; end; end;
if age14wk<836 then do; if stopfl=1 and r4hrs14>0 and
        starw<836 then do; r4hrs14=-3; end; end;
if age14wk<888 then do; if stopfl=1 and r4hrs14>0 and
        starw<888 then do; r4hrs14=-3; end; end;
if age14wk<940 then do; if stopfl=1 and r4hrs14>0 and
        starw<940 then do; r4hrs14=-3; end; end;
if age14wk<991 then do; if stopfl=1 and r4hrs14>0 and
        starw<991 then do; r4hrs14=-3; end; end;
if age14wk<1044 then do; if stopfl=1 and r4hrs14>0 and
        starw<1044 then do; r4hrs14=-3; end; end;
if age14wk<1097 then do; if stopfl=1 and r4hrs14>0 and
        starw<1097 then do; r4hrs14=-3; end; end;
if age20wk<1097 then do; if stopfl=1 and r4hrs20>0 and
        starw<1097 then do; r4hrs20=-3; end; end;
if age20wk<1097 then do; if stopfl=1 and self=0 and
        r4ehrs20>0 and starw<1097 then do; r4ehrs20=-3;
        end; end;
if age14wk<1149 then do; if stopfl=1 and r4hrs14>0 and
        starw<1149 then do; r4hrs14=-3; end; end;
if age20wk<1149 then do; if stopfl=1 and r4hrs20>0 and
        starw<1149 then do; r4hrs20=-3; end; end;
```

```
if age20wk<1149 then do; if stopfl=1 and self=0 and
        r4ehrs20>0 and starw<1149 then do; r4ehrs20=-3;
        end; end;
end;

if e200=-5 then do;
   r4hrs14=-5; r4hrs20=-5; r4ehrs20=-5;
end;

r3hrs14=tothrs14;
r3hrs20=tothrs20;
drop tothrs14 tothrs20;

/* Initialize created variables to zero. */
thrs14=0; thrs20=0;  tehrs20=0;

/* By the construction of the dli created variables, we can
simply add the two created variables from the current and
previous rounds if they are both postive. If one is positive
and one is not, then the positive value will be the total
created variable for both rounds. If neither is positive,
then the total created variable for both rounds will be 0.*/
if r4hrs14=>0 then do; thrs14=r4hrs14; end;
if r4hrs14=-3 then do; thrs14=-3; end;

/* dli is Round 3 */
if r3hrs14=>0 and r4hrs14=>0 then do;
        thrs14=r3hrs14+r4hrs14; end;
if r3hrs14=-3 then do; thrs14=-3; end;

/* dli is Round 2 */
if r2hrs14=>0 and r4hrs14=>0 and r3int=-5 then do;
        thrs14=r2hrs14+r4hrs14; end;
if r2hrs14=-3 and r3int=-5 then do; thrs14=-3; end;

/* dli is Round 1 */
if r1hrs14=>0 and r4hrs14=>0 and r3int=-5 and r2int=-5
        then do; thrs14=r1hrs14+r4hrs14; end;
if r1hrs14=-3 and r3int=-5 and r2int=-5 then do; thrs14=-
        3; end;

if r4hrs20>0 then thrs20=r4hrs20;
if r3hrs20>0 then thrs20=r3hrs20;
if r4hrs20>0 and r3hrs20>0 then do;
        thrs20=r4hrs20+r3hrs20; end;
if r4hrs20=-3 or r3hrs20=-3 then do; thrs20=-3; end;

if r4ehrs20>0 then tehrs20=r4ehrs20;
if r3hrs20>0 then tehrs20=r3hrs20;
if r4ehrs20>0 and r3hrs20>0 then do;
        tehrs20=r4ehrs20+r3hrs20; end;
if r4ehrs20=-3 or r3hrs20=-3 then do; tehrs20=-3; end;

/* For respondents younger than 20 at interview date,
        their thrs20 c.v. will be -4. */
if age20wk>intwk and intwk>0 then do; thrs20=-4;
        tehrs20=-4; end;

/* Account for non-interview cases */
if uid1=-5 then do;
   thrs14=-5; thrs20=-5; tehrs20=-5;
end;

endsas;
```

NUMBER OF JOBS HELD DURING 19XX/20XX

Variables Created: CV_TTL_JOB_ALL.80 – CV_TTL_JOB_ALL.01
CV_TTL_JOB_ET.80 – CV_TTL_JOB_ET.01
CV_TTL_JOB_SE.98 – CV_TTL_JOB_SE.01

Programs Used

This program uses **emp_begin.sas** as input (see the first page of this appendix for details).

This program calculates the number of employee-type jobs the respondent held during each calendar year. Three sets of variables are created: one for all jobs, one for employee-type jobs, and one for self-employed jobs. **See the introduction to this appendix for important information about the universes for each of these variables.** These variables are created only for respondents who have worked at least one week in the applicable calendar year.

```
array job1wks (i) wk1_1-wk1_1148;
array job2wks (i) wk2_1-wk2_1148;
array job3wks (i) wk3_1-wk3_1148;
array job4wks (i) wk4_1-wk4_1148;
array job5wks (i) wk5_1-wk5_1148;
array job6wks (i) wk6_1-wk6_1148;
array job7wks (i) wk7_1-wk7_1148;
array job8wks (i) wk8_1-wk8_1148;
array job9wks (i) wk9_1-wk9_1148;
array starw (i) starw1-starw9;
array stopw (i) stopw1-stopw9;
array starfl (i) starfl_1-starfl_9;
array stopfl (i) stopfl_1-stopfl_9;

/** Indicate if worked at least one week on a job in a
given year ** /

/* 1980 */
r4j801=0;     r4s801=0;     r4e801=0;
r4j802=0;     r4s802=0;     r4e802=0;
r4j803=0;     r4s803=0;     r4e803=0;
r4j804=0;     r4s804=0;     r4e804=0;
r4j805=0;     r4s805=0;     r4e805=0;
r4j806=0;     r4s806=0;     r4e806=0;
r4j807=0;     r4s807=0;     r4e807=0;
r4j808=0;     r4s808=0;     r4e808=0;
r4j809=0;     r4s809=0;     r4e809=0;

do i=1 to 52;
  if job1wks=-3 then do; r4j801=-3; end;
  if job1wks=-3 and self1=1 then do; r4s801=-3; end;
  if job1wks=-3 and self1=0 then do; r4e801=-3; end;
  if job2wks=-3 then do; r4j802=-3; end;
  if job2wks=-3 and self2=1 then do; r4s802=-3; end;
  if job2wks=-3 and self2=0 then do; r4e802=-3; end;
    /*and so on through*/
  if job9wks=-3 then do; r4j809=-3; end;
  if job9wks=-3 and self9=1 then do; r4s809=-3; end;
  if job9wks=-3 and self9=0 then do; r4e809=-3; end;
end;

do i=1 to 52;
  if job1wks=1 then do; r4j801=1; end;
  if job1wks=1 and self1=1 then do; r4s801=1; end;
  if job1wks=1 and self1=0 then do; r4e801=1; end;
  if job2wks=1 then do; r4j802=1; end;
  if job2wks=1 and self2=1 then do; r4s802=1; end;
  if job2wks=1 and self2=0 then do; r4e802=1; end;
    /*and so on through*/
  if job9wks=1 then do; r4j809=1; end;
  if job9wks=1 and self9=1 then do; r4s809=1; end;
  if job9wks=1 and self9=0 then do; r4e809=1; end;
end;

if r4j801 ne -3 and r4j802 ne -3 and r4j803 ne -3 and
      r4j804 ne -3 and r4j805 ne -3 and r4j806 ne -3 and
      r4j807 ne -3 and r4j808 ne -3 and r4j809 ne -3
      then do;
  r4job80=sum(r4j801, r4j802, r4j803, r4j804, r4j805,
      r4j806, r4j807, r4j808, r4j809);
end;
if r4j801=-3 or r4j802=-3 or r4j803=-3 or r4j804=-3 or
      r4j805=-3 or r4j806=-3 or r4j807=-3 or r4j808=-3
      or r4j809=-3 then do;
  r4job80=-3;
end;

if r4s801 ne -3 and r4s802 ne -3 and r4s803 ne -3 and
      r4s804 ne -3 and r4s805 ne -3 and r4s806 ne -3 and
      r4s807 ne -3 and r4s808 ne -3 and r4s809 ne -3
      then do;
  r4self80=sum(r4s801, r4s802, r4s803, r4s804, r4s805,
      r4s806, r4s807, r4s808, r4s809);
end;
if r4s801=-3 or r4s802=-3 or r4s803=-3 or r4s804=-3 or
      r4s805=-3 or r4s806=-3 or r4s807=-3 or r4s808=-3
      or r4s809=-3 then do;
  r4self80=-3;
end;

if r4e801 ne -3 and r4e802 ne -3 and r4e803 ne -3 and
      r4e804 ne -3 and r4e805 ne -3 and r4e806 ne -3
      and r4e807 ne -3 and r4e808 ne -3 and r4e809 ne -
      3 then do;
```

```
  r4emp80=sum(r4e801, r4e802, r4e803, r4e804,
      r4e805, r4e806, r4e807, r4e808, r4e809);
end;
if r4e801=-3 or r4e802=-3 or r4e803=-3 or r4e804=-3 or
      r4e805=-3 or r4e806=-3 or r4e807=-3 or r4e808=-3
      or r4e809=-3 then do;
  r4emp80=-3;
end;

/* At this point the program repeats the above code for
each year through 2001. The code is omitted here due to
space considerations. Users should contact NLS User
Services for more information. The week numbers for the
"do i" statements are as follows:
      1981   do i=53 to 104      1992   do i=627 to 679
      1982   do i=105 to 156     1993   do i=679 to 730
      1983   do i=157 to 209     1994   do i=731 to 783
      1984   do i=210 to 261     1995   do i=784 to 835
      1985   do i=262 to 313     1996   do i=836 to 887
      1986   do i=314 to 365     1997   do i=888 to 939
      1987   do i=366 to 417     1998   do i=940 to 991
      1988   do i=418 to 470     1999   do i=992 to 1044
      1989   do i=471 to 522     2000   do i=1045 to 1096
      1990   do i=523 to 574     2001   do i=1097 to 1148  */
      1991   do i=575 to 626

array self (i) self1-self9;
do i=1 to 9;
/* start date invalid */
if (stopfl=1 or starfl=1) and self=1 and r4job80>0 then
      do; r4self80=-3; r4job80=-3; end;
if (stopfl=1 or starfl=1) and self=0 and r4job80>0 then
      do; r4emp80=-3; r4job80=-3; end;
if (stopfl=1 or starfl=1) and self=1 and r4job81>0 then
      do; r4self81=-3; r4job81=-3; end;
if (stopfl=1 or starfl=1) and self=0 and r4job81>0 then
      do; r4emp81=-3; r4job81=-3; end;
  /*and so on through*/
if (stopfl=1 or starfl=1) and self=1 and r4job01>0 then
      do; r4self01=-3; r4job01=-3; end;
if (stopfl=1 or starfl=1) and self=0 and r4job01>0 then
      do; r4emp01=-3; r4job01=-3; end;
end;

array r4job (i) r4job80-r4job99 r4job00-r4job01;
array r4self (i) r4self80-r4self99 r4self00-r4self01;
array r4emp (i) r4emp80-r4emp99 r4emp00-r4emp01;

*** Include valid skips & non-interview cases;
do over r4job;
  if e200=-5 then do;
    r4job=-5; r4self=-5; r4emp=-5;
  end;
end;

/*merge in data from previous rounds*/

/* R3 variable */ array r3job r3job80-r3job99 r3job00;
/* Total cv */ array totjob totjob80-totjob99 totjob00;
do over totjob; r3job=totjob; end;

/* (2-26-01) Begin by renaming the infile c.v. variables */
array r1job (i) r1job80-r1job99 r1job00 r1job01;
array r2job (i) r2job80-r2job99 r2job00 r2job01;
array r3job (i) r3job80-r3job99 r3job00 r3job01;
array r4job (i) r4job80-r4job99 r4job00-r4job01;
array r4self (i) r4self80-r4self99 r4self00-r4self01;
array r4emp (i) r4emp80-r4emp99 r4emp00-r4emp01;
array tjob (i) tjob80-tjob99 tjob00 tjob01;
array tsjob (i) tsjob80-tsjob99 tsjob00 tsjob01;
array tejob (i) tejob80-tejob99 tejob00 tejob01;

/* Initialize created variable */
do over tjob;
  tjob=0; tsjob=0; tejob=0;
end;

/* Begin by adding together the created variables from
both rounds. Jobs that are double counted will be
subtracted off later in the program. */
do over tjob;
  if r4job>0 then do; tjob=r4job; end;
  if r4emp>0 then do; tejob=r4emp; end;
  if r4self>0 then do; tsjob=r4self; end;
  if r3job>0 and intwk>0 then do; tjob=r3job;
      tejob=r3job; end;
  if r2job>0 and r3int=-5 and intwk>0 then do;
      tjob=r2job; tejob=r2job; end;
  if r1job>0 and r2int=-5 and r3int=-5 and intwk>0 then
      do; tjob=r1job; tejob=r1job; end;
  if r3job>0 and r4job>0 then do; tjob=r3job+r4job; end;
  if r3job>0 and r4emp>0 then do; tejob=r3job+r4emp;
      end;
  if r2job>0 and r4job>0 and r3int=-5 then do;
      tjob=r2job+r4job; end;
  if r2job>0 and r4emp>0 and r3int=-5 then do;
      tejob=r2job+r4emp; end;
  if r1job>0 and r4job>0 and r2int=-5 and r3int=-5 then
      do; tjob=r1job+r4job; end;
  if r1job>0 and r4emp>0 and r2int=-5 and r3int=-5 then
      do; tejob=r1job+r4emp; end;
end;

/* Define negative values for the total created variable. */
do over tjob;
  if r4job in (-1,-2,-3) then tjob=-3;
  if r4self in (-1,-2,-3) then tsjob=-3;
  if r4emp in (-1,-2,-3) then tejob=-3;
  if r3job in (-1,-2,-3) and intwk>0 then do; tjob=-3;
      tejob=-3; end;
  if r2job in (-1,-2,-3) and r3int=-5 and intwk>0 then do;
      tjob=-3; tejob=-3; end;
  if r1job in (-1,-2,-3) and r2int=-5 and r3int=-5 and
      intwk>0 then do; tjob=-3; tejob=-3; end;
end;
```

```
/* Note there are 7 jobs in Round 1, array matched to
make programming easier */
array r1uid (i) r1uid1-r1uid9;
array r2uid (i) r2uid1-r2uid9;
array r3uid (i) r3uid1-r3uid9;
array r4uid (i) uid1-uid9;

array match1 (i) match11-match19;
array match2 (i) match21-match29;
   /*and so on through*/
array match9 (i) match91-match99;

do i=1 to 9;
  match1=0; match2=0; match3=0; match4=0;
      match5=0; match6=0; match7=0; match8=0;
      match9=0;
end;

/* Define match14 as the dummy variable that equals one
when the first job on the Round1 UID roster and the
fourth job on the Round4 roster have the same UID. */

/* If any UID from the first position in a previous round
matches with any UID in Round 4.  Need to make sure
that we find the most recent round for that job. */
do over r4uid;
  if r3uid1>0 and r4uid>0 and r3int>0 and intwk>0 then
      do;
    if r3uid1=r4uid then do; match1=3; end;
  end;
  if r2uid1>0 and r4uid>0 and match1=0 and intwk>0
      then do;
    if r2uid1=r4uid then do; match1=2; end;
  end;
  if r1uid1>0 and r4uid>0 and match1=0 and intwk>0
      then do;
    if r1uid1=r4uid then do; match1=1; end;
  end;
end;

/* If any UID from the second position in a previous
      round matches with any UID in Round 4 */
do over r4uid;
  if r3uid2>0 and r4uid>0 and r3int>0 and intwk>0 then
      do;
    if r3uid2=r4uid then do; match2=3; end;
  end;
  if r2uid2>0 and r4uid>0 and match2=0 and intwk>0
      then do;
    if r2uid2=r4uid then do; match2=2; end;
  end;
  if r1uid2>0 and r4uid>0 and match2=0 and intwk>0
      then do;
    if r1uid2=r4uid then do; match2=1; end;
  end;
end;

/*repeat the above code for UIDs in positions 3-9 in
      previous rounds*/

r4match_1=max(match11,match21,match31,match41,
      match51,match61,match71,match81,match91);
r4match_2=max(match12,match22,match32,match42,
      match52,match62,match72,match82,match92);
  /*and so on through*/
r4match_9=max(match19,match29,match39,match49,
      match59,match69,match79,match89,match99);

if match11<r4match_1 then match11=0;
if match21<r4match_1 then match21=0;
  /*and so on through*/
if match91<r4match_1 then match91=0;

if match12<r4match_2 then match12=0;
if match22<r4match_2 then match22=0;
  /*and so on through*/
if match92<r4match_2 then match92=0;

if match13<r4match_3 then match13=0;
if match23<r4match_3 then match23=0;
  /*and so on through*/
if match93<r4match_3 then match93=0;

if match14<r4match_4 then match14=0;
if match24<r4match_4 then match24=0;
  /*and so on through*/
if match94<r4match_4 then match94=0;

if match15<r4match_5 then match15=0;
if match25<r4match_5 then match25=0;
  /*and so on through*/
if match95<r4match_5 then match95=0;

if match16<r4match_6 then match16=0;
if match26<r4match_6 then match26=0;
  /*and so on through*/
if match96<r4match_6 then match96=0;

if match17<r4match_7 then match17=0;
if match27<r4match_7 then match27=0;
  /*and so on through*/
if match97<r4match_7 then match97=0;

if match18<r4match_8 then match18=0;
if match28<r4match_8 then match28=0;
  /*and so on through*/
if match98<r4match_8 then match98=0;

if match19<r4match_9 then match19=0;
if match29<r4match_9 then match29=0;
  /*and so on through*/
if match99<r4match_9 then match99=0;
```

```
/* Define "same98" as a counter variable that adds up
how many matching UIDs are in the respondent's UID
roster. This will be subtracted from the total number of
jobs created variable to avoid double counting the same
job. Begin by intializing the "same" variables to zero. */
same97=0; same98=0; same99=0; same00=0;

array r1j97 (i) r1j971-r1j979;
array r1j98 (i) r1j981-r1j989;
  /*and so on through*/
array r4j01 (i) r4j011-r4j019;

/* Consider the respondents with a Round 1 interview
date in 1997, non-interview cases only. */
if r1int_y=1997 and intwk>0 then do;
  do over match1;
    if match1=1 and r4j97=1 and r1j971=1 then do;
      same97=same97+1; end;
  end;
  do over match2;
    if match2=1 and r4j97=1 and r1j972=1 then do;
      same97=same97+1; end;
  end;
  /*and so on through*/
  do over match9;
    if match9=1 and r4j97=1 and r1j979=1 then do;
      same97=same97+1; end;
  end;
end;

/* Consider the respondents with a Round 1 interview
date in 1998, non-interview cases only. */
if r1int_y=1998 and intwk>0 then do;
  do over match1;
    if match1=1 and r4j98=1 and r1j981=1 then do;
      same98=same98+1; end;
  end;
  /*and so on through*/
  do over match9;
    if match9=1 and r4j98=1 and r1j989=1 then do;
      same98=same98+1; end;
  end;
end;

/* respondents with a Round 2 interview date in 1998. */
if r2int_y=1998 and intwk>0 then do;
  do over match1;
    if match1=2 and r4j98=1 and r2j981=1 then do;
      same98=same98+1; end;
  end;
  /*and so on through*/
  do over match9;
    if match9=2 and r4j98=1 and r2j989=1 then do;
      same98=same98+1; end;
  end;
end;

/* respondents with a Round 2 interview date in 1999. */
if r2int_y=1999 and intwk>0 then do;
  do over match1;
    if match1=2 and r4j99=1 and r2j991=1 then do;
      same99=same99+1; end;
  end;
  /*and so on through*/
  do over match9;
    if match9=2 and r4j99=1 and r2j999=1 then do;
      same99=same99+1; end;
  end;
end;

/* respondents with a Round 3 interview date in 1999. */
if r3int_y=1999 and intwk>0 then do;
  do over match1;
    if match1=3 and r4j99=1 and r3j991=1 then do;
      same99=same99+1; end;
  end;
  /*and so on through*/
  do over match9;
    if match9=3 and r4j99=1 and r3j999=1 then do;
      same99=same99+1; end;
  end;
end;

/* respondents with a Round 3 interview date in 2000. */
if r3int_y=2000 and intwk>0 then do;
  do over match1;
    if match1=3 and r4j00=1 and r3j001=1 then do;
      same00=same00+1; end;
  end;
  /*and so on through*/
  do over match9;
    if match9=3 and r4j00=1 and r3j009=1 then do;
      same00=same00+1; end;
  end;
end;

/* Now subtract the "same" count variable from the total
created variable computed above. */
if tjob97=>0 then do; tjob97=tjob97-same97; end;
if tejob97=>0 then do; tejob97=tejob97-same97; end;
if tjob98=>0 then do; tjob98=tjob98-same98; end;
if tejob98=>0 then do; tejob98=tejob98-same98; end;
if tjob99=>0 then do; tjob99=tjob99-same99; end;
if tejob99=>0 then do; tejob99=tejob99-same99; end;
if tjob00=>0 then do; tjob00=tjob00-same00; end;
if tejob00=>0 then do; tejob00=tejob00-same00; end;

/* 12-18-01 Round 4 Non-interview cases */
do over tjob;
  if uid1=-5 then do; tjob=-5; tejob=-5; tsjob=-5; end;
end;

endsas;
```

TOTAL NUMBER OF JOBS HELD AGE 14-19, NUMBER OF EMPLOYEE JOBS HELD SINCE AGE 20, AND NUMBER OF ALL JOBS HELD SINCE AGE 20

Variables Created: CV_TTL_JOBS_TEEN
CV_TTL_JOBS_ADULT_ET
CV_TTL_JOBS_ADULT_ALL

Programs Used

This program uses **emp_begin.sas** as input (see the first page of this appendix for details).

This program calculates the total number of employee-type jobs held by the respondent between the ages of 14 and 19. It then creates two variables for the total number of jobs held by the respondent since age 20: one for all jobs and one for employee-type jobs. **See the introduction to this appendix for important information about the universes for each of these variables.** The variables are only created for respondents who have worked at least one week in the applicable age range.

```
array job1wks (i) wk1_1-wk1_1148;
array job2wks (i) wk2_1-wk2_1148;
  /*and so on through*/
array job9wks (i) wk9_1-wk9_1148;
array starfl (i) starfl_1-starfl_9;
array stopfl (i) stopfl_1-stopfl_9;

** Indicate if worked at least one week on a given job
since age 14 & 20 **
job141=0;  job201=0;  ejob201=0;
job142=0;  job202=0;  ejob202=0;
  /*and so on through*/
job149=0;  job209=0;  ejob209=0;

/*Rs under 20 at int date, count jobs worked to date. */
if age14wk>0 and (age20wk-1)>1148 then do;
  do i=age14wk to 1148;
    if job1wks in (-3,1) and self1=0 then job141=1;
    if job2wks in (-3,1) and self2=0 then job142=1;
  /*and so on through*/
    if job9wks in (-3,1) and self9=0 then job149=1;
  end;
end;

/* Rs over 20 at int date, count jobs worked before age
20. */
if age14wk>0 and (age20wk-1)<=1148 then do;
  do i=age14wk to (age20wk-1);
    if job1wks in (-3,1) and self1=0 then job141=1;
    if job2wks in (-3,1) and self2=0 then job142=1;
  /*and so on through*/
    if job9wks in (-3,1) and self9=0 then job149=1;
  end;
end;

/* Rs over 20 at int date, count employee jobs worked
after age 20. */
if age14wk>0 and (age20wk-1)<=1148 then do;
  do i=age20wk to 1148;
    if job1wks in (-3,1) and self1=0 then ejob201=1;
    if job2wks in (-3,1) and self2=0 then ejob202=1;
  /*and so on through*/
    if job9wks in (-3,1) and self9=0 then ejob209=1;
  end;
end;

/*Rs over 20 at int date, count jobs worked after age 20.*/
if age14wk>0 and (age20wk-1)<=1148 then do;
  do i=age20wk to 1148;
    if job1wks in (-3,1) then job201=1;
    if job2wks in (-3,1) then job202=1;
  /*and so on through*/
    if job9wks in (-3,1) then job209=1;
  end;
end;

r4job14=sum(job141,job142,job143,job144,job145,
      job146,job147,job148,job149);
r4job20=sum(job201,job202,job203,job204,job205,
      job206,job207,job208,job209);
r4emp20=sum(ejob201,ejob202,ejob203,ejob204,
      ejob205,ejob206,ejob207,ejob208,ejob209);

* only need to assign an invalid skip to cases where the
stop date is unknown and the job starts prior to the
14/20th birthday;
if stopfl_1=1 and starfl_1 ne 1 and starw1<age14wk and
      self1=0 then do; r4job14=-3; end;
if stopfl_1=1 and starfl_1 ne 1 and starw1<age20wk and
      self1=0 and age20wk<intwk then do; r4emp20=-3;
      end;
if stopfl_1=1 and starfl_1 ne 1 and starw1<age20wk and
      age20wk<intwk then do; r4job20=-3; end;
if stopfl_2=1 and starfl_2 ne 1 and starw2<age14wk and
      self2=0 then do; r4job14=-3; end;
if stopfl_2=1 and starfl_2 ne 1 and starw2<age20wk and
      self2=0 and age20wk<intwk then do; r4emp20=-3;
      end;
if stopfl_2=1 and starfl_2 ne 1 and starw2<age20wk and
      age20wk<intwk then do; r4job20=-3; end;
```

```
  /*and so on through*/
if stopfl_9=1 and starfl_9 ne 1 and starw9<age14wk and
      self9=0 then do; r4job14=-3; end;
if stopfl_9=1 and starfl_9 ne 1 and starw9<age20wk and
      self9=0 and age20wk<intwk then do; r4emp20=-3;
      end;
if stopfl_9=1 and starfl_9 ne 1 and starw9<age20wk and
      age20wk<intwk then do; r4job20=-3; end;

if e200=-5 then do;
  r4job14=-5; r4job20=-5; r4emp20=-5;
end;

array job14 (i) job141-job149;
array job20 (i) job201-job209;
array r3j14 (i) r3j141-r3j149;
array r3j20 (i) r3j201-r3j209;

do over job14; r3j14=job14; r3j20=job20; end;
r3job14=totjob14; r3job20=totjob20;
drop totjob14 totjob20 job141-job149 job201-job209;

/* Initialize final created variable */
totjob14=0; totjob20=0; totemp20=0;

/* Begin by adding together the created variables from
both rounds. Jobs that are double counted will be
subtracted off later in the program. */
if r4job14>0 then do; totjob14=r4job14; end;
if r3job14>0 and intwk>0 then do; totjob14=r3job14;
      end;
if r2job14>0 and r3int=-5 and intwk>0 then do;
      totjob14=r2job14; end;
if r1job14>0 and r2int=-5 and r1int=-5 and intwk>0 then
      do; totjob14=r1job14; end;

if r3job14>0 and r4job14>0 then do;
      totjob14=r3job14+r4job14; end;
if r2job14>0 and r4job14>0 and r3int=-5 then do;
      totjob14=r2job14+r4job14; end;
if r1job14>0 and r4job14>0 and r2int=-5 and r3int=-5
      then do; totjob14=r1job14+r4job14; end;

if r4job20>0 then do; totjob20=r4job20; end;
if r3job20>0 and intwk>0 then do; totjob20=r3job20;
      end;
if r3job20>0 and r4job20>0 then do;
      totjob20=r3job20+r4job20; end;

if r4emp20>0 then do; totemp20=r4emp20; end;
if r3job20>0 and intwk>0 then do; totemp20=r3job20;
      end;
if r3job20>0 and r4emp20>0 then do;
      totemp20=r3job20+r4emp20; end;

/* Jobs that are worked in 1998 or 1999 have the
      possibility of being double counted since they are
      counted in each Round's created variable programs.
      Here we will try to compare the UID's to see which
      jobs are double counted. */
array r3j14 (i) r3j141-r3j149;
array r3j20 (i) r3j201-r3j209;
array job14 (i) job141-job149;
array job20 (i) job201-job209;
array r1uid (i) r1uid1-r1uid7;
array r2uid (i) r2uid1-r2uid9;
array r3uid (i) r3uid1-r3uid9;
array r4uid (i) uid1-uid9;
array match1 (i) match11-match19;
array match2 (i) match21-match29;
array match3 (i) match31-match39;
array match4 (i) match41-match49;
array match5 (i) match51-match59;
array match6 (i) match61-match69;
array match7 (i) match71-match79;
array match8 (i) match81-match89;
array match9 (i) match91-match99;

do i=1 to 9;
  match1=0; match2=0; match3=0; match4=0;
      match5=0; match6=0; match7=0; match8=0;
      match9=0;
end;

/* Define match14 as the dummy variable that equals one
      when the first job on the R1 UID roster and the
      fourth job on the R4 roster have the same UID. */

/* If any UID from the first position in dli Round
      matches with any UID in Round 4 */
do over r4uid;
  if r3uid1>0 and r4uid>0 and r3int>0 and intwk>0 then
      do;
    if r3uid1=r4uid then do; match1=1; end;
  end;
  if r2uid1>0 and r4uid>0 and r3int=-5 and intwk>0
      then do;
    if r2uid1=r4uid then do; match1=1; end;
  end;
  if r1uid1>0 and r4uid>0 and r3int=-5 and r2int=-5 and
      intwk>0 then do;
    if r1uid1=r4uid then do; match1=1; end;
  end;
end;

/* If any UID from the second position in dli Round
      matches with any UID in Round 4 */
do over r4uid;
  if r3uid2>0 and r4uid>0 and r3int>0 and intwk>0 then
      do;
    if r3uid2=r4uid then do; match2=1; end;
  end;
  if r2uid2>0 and r4uid>0 and r3int=-5 and intwk>0
      then do;
```

```
    if r2uid2=r4uid then do; match2=1; end;
  end;
  if r1uid2>0 and r4uid>0 and r3int=-5 and r2int=-5 and
      intwk>0 then do;
    if r1uid2=r4uid then do; match2=1; end;
  end;
end;

/*The above code repeates for UIDs in positions 3
through 9 in the last interview round.  Contact NLS User
Services for more information.*/

/* Define "same" as a counted variable that adds up how
      many matching UIDs are in the respondent's UID
      roster. This will be subtracted from the total
      number of jobs created variable to avoid double
      counting the same job. Begin by initializing the
      "same" variables to zero. */
same14=0;
same20=0;

do over match1;
  if match1=1 and r3j141=1 and job14=1 then do;
      same14=same14+1; end;
  if match1=1 and job14=1 and r3int=-5 then do;
      same14=same14+1; end;
  if match1=1 and r3j201=1 and job20=1 then do;
      same20=same20+1; end;
end;

do over match2;
  if match2=1 and r3j142=1 and job14=1 then do;
      same14=same14+1; end;
  if match2=1 and job14=1 and r3int=-5 then do;
      same14=same14+1; end;
  if match2=1 and r3j202=1 and job20=1 then do;
      same20=same20+1; end;
end;
/*repeat the above for match3 through match9*/

/* Now subtract the "same" count variable from the total
      created variable computed above. */
if totjob14=>0 then do; totjob14=totjob14-same14; end;
if totjob20=>0 then do; totjob20=totjob20-same20; end;
if totemp20=>0 then do; totemp20=totemp20-same20;
      end;

/* Define negative values for the total created variable. */
if r1job14 in (-1,-2,-3) and intwk>0 and r3int=-5 and
      r2int=-5 then totjob14=-3;
if r2job14 in (-1,-2,-3) and intwk>0 and r3int=-5 then
      totjob14=-3;
if r3job14 in (-1,-2,-3) and intwk>0 then totjob14=-3;
if r4job14 in (-1,-2,-3) then totjob14=-3;

if r3job20 in (-1,-2,-3) and intwk>0 then totjob20=-3;
if r4job20 in (-1,-2,-3) then totjob20=-3;

if r3job20 in (-1,-2,-3) and intwk>0 then totemp20=-3;
if r4emp20 in (-1,-2,-3) then totemp20=-3;


/* For respondents younger than 20 at interview date,
      their thrs14 c.v. will be -4. */
if age20wk>intwk and intwk>0 then do;
  totjob20=-4; totemp20=-4;
end;

if uid1=-5 then do;
  totjob14=-5; totjob20=-5; totemp20=-5;
end;

endsas;
```

Hourly Rate of Pay, Hourly Monetary Compensation, and Job Length < 13 Weeks

Variables Created: CV_HRLY_PAY
CV_HRLY_COMPENSATION
CV_JOB<13_WKS

Variables Used

Name in Program	Question Name on CD	Name in Program	Question Name on CD
e204	YEMP-204	e2260411-e2260419	YEMP-22604.01~000001-~000009
e6001-e6007	YEMP-600.01-.07	e2260410	YEMP-22604.01~000010
e192001-e192009	YEMP-19200.01-.09	e2260421-e2260429	YEMP-22604.02~000001-~000009
e207001-e207009	YEMP-20700.01-.09	e2260420	YEMP-22604.02~000010
e230001-e230009	YEMP-23000.01-.09	e2260431-e2260439	YEMP-22604.03~000001-~000009
e232001-e232003, e232007	YEMP-23200.01-.03, .07	e2260430	YEMP-22604.03~000010
e239001, e239002, e239005	YEMP-23900.01, .02, .05	e2260441-e2260449	YEMP-22604.04~000001-~000009
e239011-e239019	YEMP-23901.01-.09	e2260440	YEMP-22604.04~000010
e245011-e245016, e245018	YEMP-24501.01-.06, .08	e2260451-e2260459	YEMP-22604.05~000001-~000009
e245021-e245026, e245028	YEMP-24502.01-.06, .08	e2260450	YEMP-22604.05~000010
e245141-e245146	YEMP-24514.01-.06	e2260461-e2260469	YEMP-22604.06~000001-~000009
e24514b1	YEMP-24514B.01	e2260460	YEMP-22604.06~000010
e2451711, e2451712	YEMP-24517.01.01, .02	e2260471-e2260479	YEMP-22604.07~000001-~000009
e2680011, e2680012	YEMP-26800.01.01, .02	e2120011-e2120016	YEMP-21200.01~000001-~00006
e334001-e334007	YEMP-33400.01-.07	e2120021-e2120026	YEMP-21200.02~000001-~00006
e335001-e335005	YEMP-33500.01-.05	e2120031-e2120036	YEMP-21200.03~000001-~00006
e336001-e336005	YEMP-33600.01-.05	e2120041-e2120046	YEMP-21200.04~000001-~000006
e344021-e344027	YEMP-34402.01-.07	e2120051-e2120056	YEMP-21200.05~000001-~000006
e34402aab1-e34402aab5	YEMP-34402AAB.01-.05	e2120061-e2120066	YEMP-21200.06~000001-~000006
e34402aad1-e34402aad5	YEMP-34402AAD.01-.05	e2120071-e2120076	YEMP-21200.07~000001-~000006
e34402ab1-e34402ab5	YEMP-34402AB.01-.05	e2120081-e2120086	YEMP-21200.08~000001-~000006
e34403a1-e34403a3	YEMP-34403.01-.03	e382111-e382115	YEMP-38211.01-.05
e34404a1, e34404a2	YEMP-34404.01, .02	e38329b1-e38329b4	YEMP-38329B.01-.04
e344043	YEMP-34404.03	e38329d1-e38329d4	YEMP-38329D.01-.04
e34413c1-e34413c3	YEMP-34413C.01-.03	e38329i1, e38329i4	YEMP-38329I.01, .04
e34413e1	YEMP-34413E.01	e38329ib1, e38329ib4	YEMP-38329IB.01, .04
e3441611, e3441612	YEMP-34416.01.01, .02	e38329k1-e38329k3	YEMP-38329K.01, .03
e3442511, e3442512	YEMP-34425.01.01, .02	e3840711-e3840715	YEMP-38407.01.01-.05
e3442611, e3442612	YEMP-34426.01.01, .02	e3840721-e3840725	YEMP-38407.02.01-.05
e344281-e344287	YEMP-34428.01-.07	e3840731-e3840734	YEMP-38407.03.01-.04
e344291-e344295	YEMP-34429.01-.05	e3840741-e3840745	YEMP-38407.04.01-.05
e344301	YEMP-34430.01	e3840751	YEMP-38407.05.01
e356001-e356004	YEMP-35600.01-.04	e3841611-e3841615	YEMP-38416.01.01-.05
e361001-e361004	YEMP-36100.01-.04	e3841621-e3841625	YEMP-38416.02.01-.05
e362001-e362004	YEMP-36200.01-.04	e3841631-e3841634	YEMP-38416.03.01-.04
e2160012-e2160016	YEMP-21600.01.02-.06	e3841641-e3841643, e3841645	YEMP-38416.04.01-.03, .05
e2160022-e2160026	YEMP-21600.02.02-.06	e3841651	YEMP-38416.05.01
e2160032-e2160036	YEMP-21600.03.02-.06	e38416i11-e38416i15	YEMP-38416I.01.01-.05
e2160042-e2160046	YEMP-21600.04.02-.06	e38416i22, e38416i24	YEMP-38416I.02.02, .04
e2160052-e2160055	YEMP-21600.05.02-.05	e38416ib11-e38416ib135	YEMP-38416IB.01.01-.05
e2160062-e2160065	YEMP-21600.06.02-.05	e38416ib22, e38416ib24	YEMP-38416IB.02.02, .04
e2160072, e2160073	YEMP-21600.07.02, .03	e38416k11-e38416k15	YEMP-38416K.01.01-.05
e2160083, e2160084	YEMP-21600.08.03, .04	e38416k21, e38416k25	YEMP-38416K.02.01, .05
e2250012-e2250016	YEMP-22500.01.02-.06	e38416k31	YEMP-38416K.03.01
e2250022-e2250026	YEMP-22500.02.02-.06	e38416k45	YEMP-38416K.04.05
e2250032-e2250036	YEMP-22500.03.02-.06	e38416k51	YEMP-38416K.05.01
e2250042-e2250046	YEMP-22500.04.02-.06	e582011-e582019	YEMP-58201.01-.09
e2250052-e2250055	YEMP-22500.05.02-.05	e599001-e599009	YEMP-59900.01-.09
e2250062-e2250065	YEMP-22500.06.02-.05	e831001-e831009	YEMP-83100.01-.09
e2250072, e2250073	YEMP-22500.07.02, .03	e868001-e868007	YEMP-86800.01-.07
e2250083, e2250084	YEMP-22500.08.03, .04	e869001-e869009	YEMP-86900.01-.09
e2253012-e2253015	YEMP-22530.01.02-.05	e871001-e871004	YEMP-87100.01-.04
e2253022-e2253026	YEMP-22530.02.02-.06	e880001-e880009	YEMP-88000.01-.09

e2253032-e2253034	YEMP-22530.03.02-.04
e2253042-e2253046	YEMP-22530.04.02-.06
e2253512-e2253515	YEMP-22535.01.02-.05
e2253522-e2253526	YEMP-22535.02.02-.06
e2253532-e2253534	YEMP-22535.03.02-.04
e2253542-e2253546	YEMP-22535.04.02-.06
e2255012-e2255016	YEMP-22550.01.02-.06
e2255022-e2255026	YEMP-22550.02.02-.06
e2255032-e2255035	YEMP-22550.03.02-.05
e2255042	YEMP-22550.04.02
e2255052	YEMP-22550.05.02
e2255062, e2255065	YEMP-22550.06.02, .05
e2255072	YEMP-22550.07.02
e2260470	YEMP-22604.07~000010
e226091-e226095	YEMP-22609.01.01-.05.01
e226101-e226103	YEMP-22610.01.02- .03.02
e226111-e226113	YEMP-22611.01.03-.03.03
e226121-e226123	YEMP-22612.01.04-.03.04
e226131-e226133	YEMP-22613.01.05-.03.05
e226141, e226142	YEMP-22614.01.06, .02.06
e226151	YEMP-22615.01.07
e226161, e226163	YEMP-22616.01.08, .03.08
e226171	YEMP-22617.01.09
e226261-e226265	YEMP-22626.01.01-.05.01
e22626c2	YEMP-22626C.02.01
e22626cb2	YEMP-22626CB.02.01
e22626e1	YEMP-22626E.01.01
e226271-e226273	YEMP-22627.01.04-.03.04
e226281-e226283	YEMP-22628.01.05-.03.05
e22628c1	YEMP-22628C.01.05
e22628cb1	YEMP-22628CB.01.05
e22628e1-e22628e3	YEMP-22628E.01.05-.03.05
e226291, e226292	YEMP-22629.01.06, .02.06
e22629c2	YEMP-22629C.02.06
e22629cb2	YEMP-22629CB.02.06
e226301	YEMP-22630.01.07
e22630e1	YEMP-22630E.01.07
e226311, e226313	YEMP-22631.01.08, .03.08
e22631c1	YEMP-22631C.01.08
e22631cb1	YEMP-22631CB.01.08
e226321	YEMP-22632.01.09
e377011-e377019	YEMP-37701.01-.09
e37901b1-e37901b9	YEMP-37901B.01-.09
e3800b1-e3800b7	YEMP-38000B.01-.07
e3800b9	YEMP-38000B.09
e3800f1-e3800f7	YEMP-38000F.01-.07
e380011-e380017	YEMP-38001.01-.07
e380019	YEMP-38001.09
e380021-e380024	YEMP-38002.01-.04
e380026	YEMP-38002.06
e380031-e380034	YEMP-38003.01-.04
e380036	YEMP-38003.06
e380121-e380124	YEMP-38012.01-.04
e380126	YEMP-38012.06
e38012b1, e38012b2	YEMP-38012B.01, .02
e380131-e380137	YEMP-38013.01-.07
e380139	YEMP-38013.09
e38013b1- e38013b3	YEMP-38013B.01-.03
e380141-e380145	YEMP-38014.01-.05
e380231-e380235	YEMP-38023.01-.05
e38024b1-e38024b3	YEMP-38024B.01-.03
e38024c1-e38024c3	YEMP-38024C.01-.03
e885011-e885017	YEMP-88501.01-.07
e885021-e885027	YEMP-88502.01-.07
e885121-e885127	YEMP-88512.01-.07
e973001-e973005	YEMP-97300.01-.05
e974001-e974007	YEMP-97400.01-.07
e975001-e975004	YEMP-97500.01-.04
e982001-e982003	YEMP-98200.01-.03
e982005, e982007	YEMP-98200.05, .07
e983001-e983005	YEMP-98300.01-.05
e984021-e984027	YEMP-98402.01-.07
e98402caa1-e98402caa5	YEMP-98402CAA.01-.05
e98402cab1-e98402cab5	YEMP-98402CAB.01-.05
e98402d1-e98402d5	YEMP-98402D.01-.05
e984031-e984033	YEMP-98403.01-.03
e98404a1- e98404a3	YEMP-98404A.01-.03
e984141-e984143	YEMP-98414.01-.03
e98414aaa1, e98414aaa2	YEMP-98414AAA.01, .02
e98414aab1, e98414aab2	YEMP-98414AAB.01, .02
e98414b2, e98414b3	YEMP-98414B.02, .03
e984291-e984293	YEMP-98429.01-.03
e995001-e995005	YEMP-99500.01-.05
e1000001-e1000005	YEMP-100000.01-.05
e1001001-e1001004	YEMP-100100.01-.04
e1020512-e1020516	YEMP-100205.01.02-.06
e1020522-e1020526	YEMP-100205.02.02-.06
e1020532-e1020536	YEMP-100205.03.02-.06
e1020542-e1020546	YEMP-100205.04.02-.06
e1020552-e1020554	YEMP-100205.05.02-.04
e1020572	YEMP-100205.07.02
e1021412-e1021416	YEMP-100214.01.02-.06
e1021422-e1021426	YEMP-100214.02.02-.06
e1021432-e1021436	YEMP-100214.03.02-.06
e1021442-e1021446	YEMP-100214.04.02-.06
e1021452-e1021454	YEMP-100214.05.02-.04
e1021472	YEMP-100214.07.02
e10214aaa12-e10214aaa15	YEMP-100214AAA.01.02-.05
e10214aaa24, e10214aaa26	YEMP-100214AAA.02.04, .06
e10214aaa32, e10214aaa36	YEMP-100214AAA.03.02, .06
e10214aaa53	YEMP-100214AAA.05.03
e10214aab12-e10214aab15	YEMP-100214AAB.01.02-.05
e10214aab24, e10214aab26	YEMP-100214AAB.02.04, .06
e10214aab32, e10214aab36	YEMP-100214AAB.03.02, .06
e10214aab53	YEMP-100214AAB.05.03
e10214b12-e10214b15	YEMP-100214B.01.02-.05
e10214b22-e10214b24	YEMP-100214B.02.02-.04
e10214b32	YEMP-100214B.03.02
e10214b42	YEMP-100214B.04.02
e10214b52	YEMP-100214B.05.02
e10214b72	YEMP-100214B.07.02
e1022511-e1022519	YEMP-100225.01~000001-~000009
e1022510	YEMP-100225.01~000010
e1022521-e1022529	YEMP-100225.02~000001-~000009
e1022520	YEMP-100225.02~000010
e1022531-e1022539	YEMP-100225.03~000001-~000009
e1022530	YEMP-100225.03~000010
e1022541-e1022549	YEMP-100225.04~000001-~000009
e1022540	YEMP-100225.04.000010
e102301-e102304	YEMP-100230.01.01-.04.01
e102311-e102314	YEMP-100231.01.02-.04.02
e102321-e102323	YEMP-100232.01.03-.03.03
e102331, e102332	YEMP-100233.01.04, .02.04
e102341, e102342	YEMP-100234.01.05, .02.05

e380271, e380272
e381011-e381016
e381021-e381025
e381031-e381033
e381051-e381055
e381061-e381066
e381071-e381075
e229001-e229008
e381161-e381165
e38116aaa1
e38116aab1
e38116b1, e38116b2
e382011-e382015
e382021-e382025
YEMP-38027.01, .02
YEMP-38101.01-.06
YEMP-38102.01-.05
YEMP-38103.01-.03
YEMP-38105.01-.05
YEMP-38106.01-.06
YEMP-38107.01-.05
YEMP-22900.01-.08
YEMP-38116.01-.05
YEMP-38116AAA.01
YEMP-38116AAB.01
YEMP-38116B.01, .02
YEMP-38201.01-.05
YEMP-38202.01-.05
e102351
e102362
e102371
e102481-e102484
e10248e1
e102491, e102492
e102501, e102502
e102511
e102522
e10252e2
e102531
id
e383291- e383296
YEMP-100235.01.06
YEMP-100236.02.07
YEMP-100237.01.08
YEMP-100248.01.01-.04.01
YEMP-100248E.01.01
YEMP-100249.01.04, .02.04
YEMP-100250.01.05, .02.05
YEMP-100251.01.06
YEMP-100252.02.07
YEMP-100252E.02.07
YEMP-100253.01.08
PUBID

Codes for Created Variable

Note that hourly rate of pay is reported with two implied decimal places.

This program creates the hourly rate of pay for NLSY97 respondents. The hourly rate of pay is constructed from stop date information for respondents whose job lasted more than 13 weeks. For all other respondents the start wage is used.

In addition, this program creates an hourly monetary compensation variable for NLSY97 respondents. This variable that includes information about all compensation received by the respondent, such as tips, bonuses, commissions, overtime, etc., in the calculation. Hourly monetary compensation differs from hourly rate of pay variable, which calculates only the base pay rate. This variable is constructed from stop date information for respondents with jobs longer than 13 weeks and start date information for other.

Finally, a variable indicating whether the jobs lasted more than 13 weeks is also created. There are up to 9 jobs reported, so each variable is created for 9 jobs.

```
/* DECLARING THE ARRAYS TO BE USED LATER IN THE PROGRAM*/
array e19200 e192001-e192009;
array e22900 e229001-e229009;
array e23900 e239001-e239009;
array e34402ab e34402ab1-e34402ab9;
array e34428 e344281-e344289;
array e36200 e362001-e362009;
array e24514 e245141-e245149;
array e38014 e380141-e380149;
array e38107 e381071-e381079;
array e38027 e380271-e380279;
array e38101 e381011-e381019;
array e38201 e382011-e382019;
array e38202 e382021-e382029;
array e34400 e344001-e344009;
array e86800 e868001-e868009;
array e87800 e878001-e878009;
array e97400 e974001-e974009;
array e98402 e984021-e984029;
array e98429l e98429l1-e98429l9;
array e99500 e995001-e995009;
array e88512 e885121-e885129;
array e34404 e344041-e344049;
array e98403 e984031-e984039;
array e98414b e98414b1-e98414b9;
array e38003 e380031-e380039;
array e38329b e38329b1-e38329b9;
array e37901b e37901b1-e37901b9;
array e23000 e230001-e230009;
array e33400 e334001-e334009;
array e33600 e336001-e336009;
array e34430 e344301-e344309;
array e20700 e207001-e207009;
array e24514b e24514b1-e24514b9;
array e38023 e380231-e380239;
array e38116 e381161-e381169;
array e23901 e239011-e239019;
array e38102 e381021-e381029;
array e38105 e381051-e381059;
array e38211 e382111-e382119;
array e38313 e383131-e383139;
array e86900 e869001-e869009;
array e88000 e880001-e880009;
array e97500 e975001-e975009;
array e98402d e98402d1-e98402d9;
array e100000 e1000001-e1000009;
array e88501 e885011-e885019;
array e88512b e88512b1-e88512b9;
array e34413c e34413c1-e34413c9;
array e98404a e98404a1-e98404a9;
array e38001 e380011-e380019;
array e38012 e380121-e380129;
array e38329d e38329d1-e38329d9;
array e59900 e599001-e599009;
array e23200 e232001-e232009;
array e34402 e344021-e344029;
array e33500 e335001-e335009;
array e36100 e361001-e361009;
array e24501 e245011-e245019;
array e38013 e380131-e380139;
array e38106 e381061-e381069;
array e3800b e3800b1-e3800b9;
array e3800f e3800f1-e3800f9;
array e38103 e381031-e381039;
array e24502 e245021-e245029;
array e38211b e38211b1-e38211b9;
array e83100 e831001-e831009;
array e87100 e871001-e871009;
array e97300 e973001-e973009;
array e98300 e983001-e983009;
array e98429 e984291-e984299;
array e100100 e1001001-e1001009;
array e88502 e885021-e885029;
array e34403 e344031-e344039;
array e34413e e34413e1-e34413e9;
array e98414 e984141-e984149;
array e38002 e380021-e380029;
array e38012b e38012b1-e38012b9;
array e38329 e383291-e383299;
```

```
array e38330 e383301-e383309;
array e35600 e356001-e356009;
array e22609 e226091-e226099;
array e22610 e226101-e226109;
array e22611 e226111-e226119;
array e22612 e226121-e226129;
array e22613 e226131-e226139;
array e22614 e226141-e226149;
array e22615 e226151-e226159;
array e22616 e226161-e226269;
array e22617 e226171-e226179;
array e22626 e226261-e226269;
array e22627 e226271-e226279;
array e22628 e226281-e226289;
array e22629 e226291-e226299;
array e22630 e226301-e226309;
array e22631 e226311-e226319;
array e22632 e226321-e226329;
array e100230 e102301-e102309;
array e100231 e102311-e102319;
array e100232 e102321-e102329;
array e100233 e102331-e102339;
array e100234 e102341-e102349;
array e100235 e102351-e102359;
array e100236 e102361-e102369;
array e100237 e102371-e102379;
array e100239 e102391-e102399;
array e100248 e102481-e102489;
array e100249 e102491-e102499;
array e100250 e102501-e102509;
array e100251 e102511-e102519;
array e100252 e102521-e102529;
array e100253 e102531-e102539;
array e100254 e102541-e102549;
array e37701 e377011-e377019;
array e58401 e584011-e584019;
array e58201 e582011-e582019;
array euid euid1-euid9;
array e38116b e38116b1-e38116b9;
array e34402aab e34402aab1-e34402aab9;
array e34402aad e34402aad1-e34402aad9;
array pay pay01-pay09;
array e98402caa e98402caa1-e98402caa9;
array e98402cab e98402cab1-e98402cab9;
array e22626c e22626c1-e22626c9;
array e22626cb e22626cb1-e22626cb9;
array e22626e e22626e1-e22626e9;
array e100248c e100248c1-e100248c9;
array e100248cb e100248cb1-e100248cb9;
array e100248e e100248e1-e100248e9;
array e24514aab e24514aab1-e24514aab9;
array e24514aabb e24514aabb1-e24514aabb9;
array e34413cc e34413cc1-e34413cc9;
array e34413cd e34413cd1-e34413cd9;
array e88512aaa e88512aaa1-e88512aaa9;
array e88512aab e88512aab1-e88512aab9;
array e98414aaa e98414aaa1-e98414aaa9;
array e98414aab e98414aab1-e98414aab9;
array e22627c e22627c1-e22627c9;
array e22627cb e22627cb1-e22627cb9;
array e22627e e22627e1-e22627e9;
array e100249c e100249c1-e100249c9;
array e100249cb e100249cb1-e100249cb9;
array e100249e e100249e1-e100249e9;
array e22628c e22628c1-e22628c9;
array e22628cb e22628cb1-e22628cb9;
array e22628e e22628e1-e22628e9;
array e100250c e100250c1-e100250c9;
array e100250cb e100250cb1-e100250cb9;
array e100250e e100250e1-e100250e9;
array e22629c e22629c1-e22629c9;
array e22629cb e22629cb1-e22629cb9;
array e22629e e22629e1-e22629e9;
array e100251c e100251c1-e100251c9;
array e100251cb e100251cb1-e100251cb9;
array e100251e e100251e1-e100251e9;
array e22630c e22630c1-e22630c9;
array e22630cb e22630cb1-e22630cb9;
array e22630e e22630e1-e22630e9;
array e100252c e100252c1-e100252c9;
array e100252cb e100252cb1-e100252cb9;
array e100252e e100252e1-e100252e9;
array e22631c e22631c1-e22631c9;
array e22631cb e22631cb1-e22631cb9;
array e22631e e22631e1-e22631e9;
array e100253c e100253c1-e100253c9;
array e100253cb e100253cb1-e100253cb9;
array e100253e e100253e1-e100253e9;
array e22632c e22632c1-e22632c9;
array e22632cb e22632cb1-e22632cb9;
array e22632e e22632e1-e22632e9;
array e100254c e100254c1-e100254c9;
array e100254cb e100254cb1-e100254cb9;
array e100254e e100254e1-e100254e9;
array e38024b e38024b1-e38024b9;
array e38024c e38024c1-e38024c9;
array endhr1 endhr11-endhr19;
array endhr2 endhr21-endhr29;
array e38116aaa e38116aaa1-e38116aaa9;
array e38116aab e38116aab1-e38116aab9;
array e38012aaa e38012aaa1-e38012aaa9;
array e38012aab e38012aab1-e38012aab9;
array e38211aaa e38211aaa1-e38211aaa9;
array e38211aab e38211aab1-e38211aab9;
array e38329g e38329g1-e38329g9;
array e38329i e38329i1-e38329i9;
array e38329ib e38329ib1-e38329ib9;
array e38329k e38329k1-e38329k9;

array e225001 e2250012 e2250022 e2250032 e2250042 e2250052 e2250062 e2250072 e2250082 e2250092;
array e225002 e2250013 e2250023 e2250033 e2250043 e2250053 e2250063 e2250073 e2250083 e2250093;
array e225003 e2250014 e2250024 e2250034 e2250044 e2250054 e2250064 e2250074 e2250084 e2250094;
array e225004 e2250015 e2250025 e2250035 e2250045 e2250055 e2250065 e2250075 e2250085 e2250095;
array e225005 e2250016 e2250026 e2250036 e2250046 e2250056 e2250066 e2250076 e2250086 e2250096;

array e225301 e2253012 e2253022 e2253032 e2253042 e2253052 e2253062 e2253072 e2253082 e2253092;
array e225302 e2253013 e2253023 e2253033 e2253043 e2253053 e2253063 e2253073 e2253083 e2253093;
```

```
array e225303 e2253014 e2253024 e2253034 e2253044 e2253054 e2253064 e2253074 e2253084 e2253094;
array e225304 e2253015 e2253025 e2253035 e2253045 e2253055 e2253065 e2253075 e2253085 e2253095;
array e225305 e2253016 e2253026 e2253036 e2253046 e2253056 e2253066 e2253076 e2253086 e2253096;

array e225351 e2253512 e2253522 e2253532 e2253542 e2253552 e2253562 e2253572 e2253582 e2253592;
array e225352 e2253513 e2253523 e2253533 e2253543 e2253553 e2253563 e2253573 e2253583 e2253593;
array e225353 e2253514 e2253524 e2253534 e2253544 e2253554 e2253564 e2253574 e2253584 e2253594;
array e225354 e2253515 e2253525 e2253535 e2253545 e2253555 e2253565 e2253575 e2253585 e2253595;
array e225355 e2253516 e2253526 e2253536 e2253546 e2253556 e2253566 e2253576 e2253586 e2253596;

array e225501 e2255012 e2255022 e2255032 e2255042 e2255052 e2255062 e2255072 e2255082 e2255092;
array e225502 e2255013 e2255023 e2255033 e2255043 e2255053 e2255063 e2255073 e2255083 e2255093;
array e225503 e2255014 e2255024 e2255034 e2255044 e2255054 e2255064 e2255074 e2255084 e2255094;
array e225504 e2255015 e2255025 e2255035 e2255045 e2255055 e2255065 e2255075 e2255085 e2255095;
array e225505 e2255016 e2255026 e2255036 e2255046 e2255056 e2255066 e2255076 e2255086 e2255096;

array e216001 e2160012 e2160022 e2160032 e2160042 e2160052 e2160062 e2160072 e2160082 e2160092;
array e216002 e2160013 e2160023 e2160033 e2160043 e2160053 e2160063 e2160073 e2160083 e2160093;
array e216003 e2160014 e2160024 e2160034 e2160044 e2160054 e2160064 e2160074 e2160084 e2160094;
array e216004 e2160015 e2160025 e2160035 e2160045 e2160055 e2160065 e2160075 e2160085 e2160095;
array e216005 e2160016 e2160026 e2160036 e2160046 e2160056 e2160066 e2160076 e2160086 e2160096;

array e212001 e2120011 e2120021 e2120031 e2120041 e2120051 e2120061 e2120071 e2120081 e2120091;
array e212002 e2120012 e2120022 e2120032 e2120042 e2120052 e2120062 e2120072 e2120082 e2120092;
array e212003 e2120013 e2120023 e2120033 e2120043 e2120053 e2120063 e2120073 e2120083 e2120093;
array e212004 e2120014 e2120024 e2120034 e2120044 e2120054 e2120064 e2120074 e2120084 e2120094;
array e212005 e2120015 e2120025 e2120035 e2120045 e2120055 e2120065 e2120075 e2120085 e2120095;
array e212006 e2120016 e2120026 e2120036 e2120046 e2120056 e2120066 e2120076 e2120086 e2120096;

array e384161 e3841611 e3841621 e3841631 e3841641 e3841651 e3841661 e3841671 e3841681 e3841691;
array e384162 e3841612 e3841622 e3841632 e3841642 e3841652 e3841662 e3841672 e3841682 e3841692;
array e384163 e3841613 e3841623 e3841633 e3841643 e3841653 e3841663 e3841673 e3841683 e3841693;
array e384164 e3841614 e3841624 e3841634 e3841644 e3841654 e3841664 e3841674 e3841684 e3841694;
array e384165 e3841615 e3841625 e3841635 e3841645 e3841655 e3841665 e3841675 e3841685 e3841695;

array e38416g1 e38416g11 e38416g21 e38416g31 e38416g41 e38416g51 e38416g61 e38416g71 e38416g81 e38416g91;
array e38416g2 e38416g12 e38416g22 e38416g32 e38416g42 e38416g52 e38416g62 e38416g72 e38416g82 e38416g92;
array e38416g3 e38416g13 e38416g23 e38416g33 e38416g43 e38416g53 e38416g63 e38416g73 e38416g83 e38416g93;
array e38416g4 e38416g14 e38416g24 e38416g34 e38416g44 e38416g54 e38416g64 e38416g74 e38416g84 e38416g94;
array e38416g5 e38416g15 e38416g25 e38416g35 e38416g45 e38416g55 e38416g65 e38416g75 e38416g85 e38416g95;

array e38416i1 e38416i11 e38416i21 e38416i31 e38416i41 e38416i51 e38416i61 e38416i71 e38416i81 e38416i91;
array e38416i2 e38416i12 e38416i22 e38416i32 e38416i42 e38416i52 e38416i62 e38416i72 e38416i82 e38416i92;
array e38416i3 e38416i13 e38416i23 e38416i33 e38416i43 e38416i53 e38416i63 e38416i73 e38416i83 e38416i93;
array e38416i4 e38416i14 e38416i24 e38416i34 e38416i44 e38416i54 e38416i64 e38416i74 e38416i84 e38416i94;
array e38416i5 e38416i15 e38416i25 e38416i35 e38416i45 e38416i55 e38416i65 e38416i75 e38416i85 e38416i95;

array e38416ib1 e38416ib11 e38416ib21 e38416ib31 e38416ib41 e38416ib51 e38416ib61 e38416ib71 e38416ib81
        e38416ib91;
array e38416ib2 e38416ib12 e38416ib22 e38416ib32 e38416ib42 e38416ib52 e38416ib62 e38416ib72 e38416ib82
        e38416ib92;
array e38416ib3 e38416ib13 e38416ib23 e38416ib33 e38416ib43 e38416ib53 e38416ib63 e38416ib73 e38416ib83
        e38416ib93;
array e38416ib4 e38416ib14 e38416ib24 e38416ib34 e38416ib44 e38416ib54 e38416ib64 e38416ib74 e38416ib84
        e38416ib94;
array e38416ib5 e38416ib15 e38416ib25 e38416ib35 e38416ib45 e38416ib55 e38416ib65 e38416ib75 e38416ib85
        e38416ib95;
```

```
array e38416k1 e38416k11 e38416k21 e38416k31 e38416k41 e38416k51 e38416k61 e38416k71 e38416k81
        e38416k91;
array e38416k2 e38416k12 e38416k22 e38416k32 e38416k42 e38416k52 e38416k62 e38416k72 e38416k82
        e38416k92;
array e38416k3 e38416k13 e38416k23 e38416k33 e38416k43 e38416k53 e38416k63 e38416k73 e38416k83
        e38416k93;
array e38416k4 e38416k14 e38416k24 e38416k34 e38416k44 e38416k54 e38416k64 e38416k74 e38416k84
        e38416k94;
array e38416k5 e38416k15 e38416k25 e38416k35 e38416k45 e38416k55 e38416k65 e38416k75 e38416k85
        e38416k95;

array e384071 e3840711 e3840721 e3840731 e3840741 e3840751 e3840761 e3840771 e3840781 e3840791;
array e384072 e3840712 e3840722 e3840732 e3840742 e3840752 e3840762 e3840772 e3840782 e3840792;
array e384073 e3840713 e3840723 e3840733 e3840743 e3840753 e3840763 e3840773 e3840783 e3840793;
array e384074 e3840714 e3840724 e3840734 e3840744 e3840754 e3840764 e3840774 e3840784 e3840794;
array e384075 e3840715 e3840725 e3840735 e3840745 e3840755 e3840765 e3840775 e3840785 e3840795;

array e102051 e1020512 e1020522 e1020532 e1020542 e1020552 e1020562 e1020572 e1020582 e1020592;
array e102052 e1020513 e1020523 e1020533 e1020543 e1020553 e1020563 e1020573 e1020583 e1020593;
array e102053 e1020514 e1020524 e1020534 e1020544 e1020554 e1020564 e1020574 e1020584 e1020594;
array e102054 e1020515 e1020525 e1020535 e1020545 e1020555 e1020565 e1020575 e1020585 e1020595;
array e102055 e1020516 e1020526 e1020536 e1020546 e1020556 e1020566 e1020576 e1020586 e1020596;

array e102141 e1021412 e1021422 e1021432 e1021442 e1021452 e1021462 e1021472 e1021482 e1021492;
array e102142 e1021413 e1021423 e1021433 e1021443 e1021453 e1021463 e1021473 e1021483 e1021493;
array e102143 e1021414 e1021424 e1021434 e1021444 e1021454 e1021464 e1021474 e1021484 e1021494;
array e102144 e1021415 e1021425 e1021435 e1021445 e1021455 e1021465 e1021475 e1021485 e1021495;
array e102145 e1021416 e1021426 e1021436 e1021446 e1021456 e1021466 e1021476 e1021486 e1021496;

array e10214aaa1 e10214aaa12 e10214aaa22 e10214aaa32 e10214aaa42 e10214aaa52 e10214aaa62 e10214aaa72
        e10214aaa82 e10214aaa92;
array e10214aaa2 e10214aaa13 e10214aaa23 e10214aaa33 e10214aaa43 e10214aaa53 e10214aaa63 e10214aaa73
        e10214aaa83 e10214aaa93;
array e10214aaa3 e10214aaa14 e10214aaa24 e10214aaa34 e10214aaa44 e10214aaa54 e10214aaa64 e10214aaa74
        e10214aaa84 e10214aaa94;
array e10214aaa4 e10214aaa15 e10214aaa25 e10214aaa35 e10214aaa45 e10214aaa55 e10214aaa65 e10214aaa75
        e10214aaa85 e10214aaa95;
array e10214aaa5 e10214aaa16 e10214aaa26 e10214aaa36 e10214aaa46 e10214aaa56 e10214aaa66 e10214aaa76
        e10214aaa86 e10214aaa96;

array e10214aab1 e10214aab12 e10214aab22 e10214aab32 e10214aab42 e10214aab52 e10214aab62 e10214aab72
        e10214aab82 e10214aab92;
array e10214aab2 e10214aab13 e10214aab23 e10214aab33 e10214aab43 e10214aab53 e10214aab63 e10214aab73
        e10214aab83 e10214aab93;
array e10214aab3 e10214aab14 e10214aab24 e10214aab34 e10214aab44 e10214aab54 e10214aab64 e10214aab74
        e10214aab84 e10214aab94;
array e10214aab4 e10214aab15 e10214aab25 e10214aab35 e10214aab45 e10214aab55 e10214aab65 e10214aab75
        e10214aab85 e10214aab95;
array e10214aab5 e10214aab16 e10214aab26 e10214aab36 e10214aab46 e10214aab56 e10214aab66 e10214aab76
        e10214aab86 e10214aab96;

array e10214b1 e10214b12 e10214b22 e10214b32 e10214b42 e10214b52 e10214b62 e10214b72 e10214b82
        e10214b92;
array e10214b2 e10214b13 e10214b23 e10214b33 e10214b43 e10214b53 e10214b63 e10214b73 e10214b83
        e10214b93;
```

```
array e10214b3 e10214b14 e10214b24 e10214b34 e10214b44 e10214b54 e10214b64 e10214b74 e10214b84
        e10214b94;
array e10214b4 e10214b15 e10214b25 e10214b35 e10214b45 e10214b55 e10214b65 e10214b75 e10214b85
        e10214b95;
array e10214b5 e10214b16 e10214b26 e10214b36 e10214b46 e10214b56 e10214b66 e10214b76 e10214b86
        e10214b96;

array e226041 e2260411 e2260421 e2260431 e2260441 e2260451 e2260461 e2260471 e2260481 e2260491;
array e226042 e2260412 e2260422 e2260432 e2260442 e2260452 e2260462 e2260472 e2260482 e2260492;
array e226043 e2260413 e2260423 e2260433 e2260443 e2260453 e2260463 e2260473 e2260083 e2260493;
array e226044 e2260414 e2260424 e2260434 e2260444 e2260454 e2260464 e2260474 e2260484 e2260494;
array e226045 e2260415 e2260425 e2260435 e2260445 e2260455 e2260465 e2260475 e2260485 e2260495;
array e226046 e2260416 e2260426 e2260436 e2260446 e2260456 e2260466 e2260476 e2260486 e2260496;
array e226047 e2260417 e2260427 e2260437 e2260447 e2260457 e2260467 e2260477 e2260487 e2260497;
array e226048 e2260418 e2260428 e2260438 e2260448 e2260458 e2260468 e2260478 e2260088 e2260498;
array e226049 e2260419 e2260429 e2260439 e2260449 e2260459 e2260469 e2260479 e2260489 e2260499;
array e226040 e2260410 e2260420 e2260430 e2260440 e2260450 e2260460 e2260470 e2260480 e2260490;

array e102251 e1022511 e1022521 e1022531 e1022541 e1022551 e1022561 e1022571 e1022581 e1022591;
array e102252 e1022512 e1022522 e1022532 e1022542 e1022552 e1022562 e1022572 e1022582 e1022592;
array e102253 e1022513 e1022523 e1022533 e1022543 e1022553 e1022563 e1022573 e2260083 e1022593;
array e102254 e1022514 e1022524 e1022534 e1022544 e1022554 e1022564 e1022574 e1022584 e1022594;
array e102255 e1022515 e1022525 e1022535 e1022545 e1022555 e1022565 e1022575 e1022585 e1022595;
array e102256 e1022516 e1022526 e1022536 e1022546 e1022556 e1022566 e1022576 e1022586 e1022596;
array e102257 e1022517 e1022527 e1022537 e1022547 e1022557 e1022567 e1022577 e1022587 e1022597;
array e102258 e1022518 e1022528 e1022538 e1022548 e1022558 e1022568 e1022578 e2260088 e1022598;
array e102259 e1022519 e1022529 e1022539 e1022549 e1022559 e1022569 e1022579 e1022589 e1022599;
array e102250 e1022510 e1022520 e1022530 e1022540 e1022550 e1022560 e1022570 e1022580 e1022590;


/***** We create the hourly wages.  First, the start wages, then the end wages ***************/
/***** Start wages for the youth ************************************************************/

/* For each time unit, the information could either be given in the earlier part (from e19200) or in the latet part (from
        e83100). And it also depends on if the respondent have other compensation or not. */
/*Set up the hourly wage for youths who report an hourly wage*/
hrwage01=-4;    hrwage02=-4;    hrwage03=-4;    hrwage04=-4;    hrwage05=-4;
hrwage06=-4;    hrwage07=-4;    hrwage08=-4;    hrwage09=-4;

array hrwage hrwage01 hrwage02 hrwage03 hrwage04 hrwage05 hrwage06 hrwage07 hrwage08 hrwage09;

do i=1 to 9;

  if e19200[i]=1 then do;
    if (e22900[i]>=0) then hrwage[i]=e22900[i];
    if (e23000[i]>=0) then hrwage[i]=e23000[i];
    if (e22900[i]=-2 or e23000[i]=-2) then hrwage[i]=e23200[i];
    if (e22900[i]=-3 or e23000[i]=-3) then hrwage[i]=e23200[i];
    if (e22900[i]=-1 or e23000[i]=-1) then hrwage[i]=-1;
    if e23900[i]>=0 then hrwage[i]=e23900[i];
  end;

  if e83100[i]=1 then do;
    if (e86800[i]>=0) then hrwage[i]=e86800[i];
    if (e86900[i]>=0) then hrwage[i]=e86900[i];
    if (e86800[i]=-2 or e86900[i]=-2) then hrwage[i]=e87100[i];
    if (e86800[i]=-3 or e86900[i]=-3) then hrwage[i]=e87100[i];
```

```
    if (e86800[i]=-1 or e86900[i]=-1) then hrwage[i]=-1;
    if e87800[i]>=0 then hrwage[i]=e87800[i];
  end;

end;

/*Set up the hourly wage for youths who report their wage in daily units.*/
daily01=-4;        daily02=-4;        daily03=-4;        daily04=-4;        daily05=-4;
daily06=-4;        daily07=-4;        daily08=-4;        daily09=-4;

array daily daily01 daily02 daily03 daily04 daily05 daily06 daily07 daily08 daily09;

do i=1 to 9;

/* daily start wage divided by the number of hours worked per week*/

if e19200[i]=2 then do;

/*no compensation*/
   if (e33400[i]>=0 and e34402[i]>0 and e34402ab[i]>0) then daily[i]=(e33400[i]*e34402ab[i]/e34402[i]);
   if (e33400[i]=-2 and e33600[i]>=0 and e34402[i]>0 and e34402ab[i]>0) then
            daily[i]=(e33600[i]*e34402ab[i]/e34402[i]);
   if e34400[i]>=0 and e34402[i]>0 and e34402ab[i]>0 then daily[i]=e34400[i]*e34402ab[i]/e34402[i];
   /* missing value*/
   if -4<e34402[i]<0 then daily[i]=e34402[i];

/*received compensation*/
  /*without overtime*/
  if (e33500[i]>=0 and e34402[i]>0 and e34402ab[i]>0) then daily[i]=(e33500[i]*e34402ab[i]/e34402[i]);
  if (e33500[i]=-2 and e33600[i]>=0 and e34402[i]>0 and e34402ab[i]>0) then
            daily[i]=(e33600[i]*e34402ab[i]/e34402[i]);
  if e34400[i]>=0 and e34402[i]>0 and e34402ab[i]>0 then daily[i]=e34400[i]*e34402ab[i]/e34402[i];
     /*missing value*/
  if -4<e34402[i]<0 then daily[i]=e34402[i];
  if (e33400[i]>=0 or e33500[i]>=0 or e33600[i]>=0 or e34400[i]>=0) and e34402[i]=0 then daily[i]=-3;

  if -4<e34402ab[i]<0 then daily[i]=e34402ab[i];
  if e34402ab[i]=0 then daily[i]=-3;

  /*with overtime*/
   if (e33500[i]>=0 and e34428[i]>0 and e34430[i]>0) then daily[i]=(e33500[i]*e34430[i]/e34428[i]);
   if (e33500[i]=-2 and e33600[i]>=0 and e34428[i]>0 and e34430[i]>0) then daily[i]=(e33600[i]*e34430[i]/e34428[i]);
   if e34400[i]>=0 and e34428[i]>0 and e34430[i]>0 then daily[i]=e34400[i]*e34430[i]/e34428[i];
     /*missing value*/
   if -4<e34428[i]<0 then daily[i]=e34428[i];
   if (e33500[i]>=0 or e33600[i]>=0 or e34400[i]>=0) and e34428[i]=0 then daily[i]=-3;
   if -4<e34430[i]<0 then daily[i]=e34430[i];
   if e34430[i]=0 then daily[i]=-3;

/*if still paid hourly...*/
   if e36200[i]>=0 then daily[i]=e36200[i];
   if e36100[i]>=0 then daily[i]=e36100[i];

end;

 if e83100[i]=2 then do;
```

```
/*no compensation*/
   if (e97300[i]>=0 and e98402[i]>0 and e98402d[i]>0) then daily[i]=(e97300[i]*e98402d[i]/e98402[i]);
   if (e97300[i]=-2 and e97500[i]>=0 and e98402[i]>0 and e98402d[i]>0) then
            daily[i]=(e97500[i]*e98402d[i]/e98402[i]);
   if e98300[i]>=0 and e98402[i]>0 and e98402d[i]>0 then daily[i]=e98300[i]*e98402d[i]/e98402[i];
   /* missing value*/
   if -4<e98402[i]<0 then daily[i]=e98402[i];

/*received compensation*/
  /*with overtime */
   if (e97400[i]>=0 and e98429[i]>0 and e98429l[i]>0) then daily[i]=(e97400[i]*e98429l[i]/e98429[i]);
   if (e97400[i]=-2 and e97500[i]>=0 and e98429[i]>0 and e98429l[i]>0) then daily[i]=(e97500[i]*e98429l[i]/e98429[i]);
   if e98300[i]>=0 and e98429[i]>0 and e98429l[i]>0 then daily[i]=e98300[i]*e98429l[i]/e98429[i];
   /*missing value*/
   if -4<e98429[i]<0 then daily[i]=e98429[i];
   if (e97400[i]>=0 or e97500[i]>=0 or e98300[i]>=0) and e98429[i]=0 then daily[i]=-3;
   if -4<e98429l[i]<0 then daily[i]=e98429l[i];
   if e98429l[i]=0 then daily[i]=-3;

  /* without overtime */
   if (e97400[i]>=0 and e98402[i]>0 and e98402d[i]>0) then daily[i]=(e97400[i]*e98402d[i]/e98402[i]);
   if (e97400[i]=-2 and e97500[i]>=0 and e98402[i]>0 and e98402d[i]>0) then
            daily[i]=(e97500[i]*e98402d[i]/e98402[i]);
   if e98300[i]>=0 and e98402[i]>0 and e98402d[i]>0 then daily[i]=e98300[i]*e98402d[i]/e98402[i];
   /*missing value*/
   if -4<e98402[i]<0 then daily[i]=e98402[i];
   if (e97300[i]>=0 or e97400[i]>=0 or e97500[i]>=0 or e98300[i]>=0) and e98402[i]=0 then daily[i]=-3;
   if -4<e98402d[i]<0 then daily[i]=e98402d[i];
   if e98402d[i]=0 then daily[i]=-3;

/*if still paid hourly...*/
   if e100100[i]>=0 then daily[i]=e100100[i];
   if e100000[i]>=0 then daily[i]=e100000[i];

end;
end;

/*set up the hourly wage for youths who report their wage in weekly units.*/
weekly01=-4;      weekly02=-4;      weekly03=-4;      weekly04=-4;      weekly05=-4;
weekly06=-4;      weekly07=-4;      weekly08=-4;      weekly09=-4;

array weekly weekly01 weekly02 weekly03 weekly04 weekly05 weekly06 weekly07 weekly08 weekly09;

do i=1 to 9;

/* weekly start wage divided by the number of hours worked per week*/

if e19200[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28, 999,-2) then do;

/*no compensation*/
   if (e33400[i]>=0 and e34402[i]>0) then weekly[i]=(e33400[i]/e34402[i]);
   if (e33400[i]=-2 and e33600[i]>=0 and e34402[i]>0) then weekly[i]=(e33600[i]/e34402[i]);
   if e34400[i]>=0 and e34402[i]>0 then weekly[i]=e34400[i]/e34402[i];
   /*missing value*/
   if -4<e34402[i]<0 then weekly[i]=e34402[i];

/*received compensation*/
```

```
  /*with overtime*/
   if (e33500[i]>=0 and e34428[i]>0) then weekly[i]=(e33500[i]/e34428[i]);
   if (e33500[i]=-2 and e33600[i]>=0 and e34428[i]>0) then weekly[i]=(e33600[i]/e34428[i]);
   if e34400[i]>=0 and e34428[i]>0 then weekly[i]=e34400[i]/e34428[i];
   /*missing value*/
   if -4<e34428[i]<0 then weekly[i]=e34428[i];
   if (e33500[i]>=0 or e33600[i]>=0 or e34400[i]>=0) and e34428[i]=0 then weekly[i]=-3;

  /*without overtime*/
   if (e33500[i]>=0 and e34402[i]>0) then weekly[i]=(e33500[i]/e34402[i]);
   if (e33500[i]=-2 and e33600[i]>=0 and e34402[i]>0) then weekly[i]=(e33600[i]/e34402[i]);
   if e34400[i]>=0 and e34402[i]>0 then weekly[i]=e34400[i]/e34402[i];
   /*missing value*/
   if -4<e34402[i]<0 then weekly[i]=e34402[i];
   if (e33400[i]>=0 or e33500[i]>=0 or e33600[i]>=0 or e34400[i]>=0) and e34402[i]=0 then weekly[i]=-3;


/*if still paid hourly...*/
   if e36200[i]>=0 then weekly[i]=e36200[i];
   if e36100[i]>=0 then weekly[i]=e36100[i];

end;

 if e83100[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28, 999,-2) then do;

/*no compensation*/
   if (e97300[i]>=0 and e98402[i]>0) then weekly[i]=(e97300[i]/e98402[i]);
   if (e97300[i]=-2 and e97500[i]>=0 and e98402[i]>0) then weekly[i]=(e97500[i]/e98402[i]);
   if e98300[i]>=0 and e98402[i]>0 then weekly[i]=e98300[i]/e98402[i];
   /*missing value*/
   if -4<e98402[i]<0 then weekly[i]=e98402[i];

/*received compensation*/
  /*with overtime*/
   if (e97400[i]>=0 and e98429[i]>0) then weekly[i]=(e97400[i]/e98429[i]);
   if (e97400[i]=-2 and e97500[i]>=0 and e98429[i]>0) then weekly[i]=(e97500[i]/e98429[i]);
   if e98300[i]>=0 and e98429[i]>0 then weekly[i]=e98300[i]/e98429[i];
   /*missing value*/
   if -4<e98429[i]<0 then weekly[i]=e98429[i];
   if (e97400[i]>=0 or e97500[i]>=0 or e98300[i]>=0) and e98429[i]=0 then weekly[i]=-3;

  /*without overtime*/
   if (e97400[i]>=0 and e98402[i]>0) then weekly[i]=(e97400[i]/e98402[i]);
   if (e97400[i]=-2 and e97500[i]>=0 and e98402[i]>0) then weekly[i]=(e97500[i]/e98402[i]);
   if e98300[i]>=0 and e98402[i]>0 then weekly[i]=e98300[i]/e98402[i];
   /*missing value*/
   if -4<e98402[i]<0 then weekly[i]=e98402[i];
   if (e97300[i]>=0 or e97400[i]>=0 or e97500[i]>=0 or e98300[i]>=0) and e98402[i]=0 then weekly[i]=-3;

/*if still paid hourly...*/
   if e100100[i]>=0 then weekly[i]=e100100[i];
   if e100000[i]>=0 then weekly[i]=e100000[i];

end;
end;

/*set up the hourly wage for youths who report their wage in biweekly units.*/
```

```
biwkly01=-4;      biwkly02=-4;      biwkly03=-4;      biwkly04=-4;      biwkly05=-4;
biwkly06=-4;      biwkly07=-4;      biwkly08=-4;      biwkly09=-4;

array biwkly biwkly01 biwkly02 biwkly03 biwkly04 biwkly05 biwkly06 biwkly07 biwkly08 biwkly09;

do i=1 to 9;

/* biwkly start wage divided by the number of hours worked per week*/

if e19200[i]=4 then do;

/*no compensation*/
   if (e33400[i]>=0 and e34402[i]>0) then biwkly[i]=e33400[i]/(2*e34402[i]);
   if (e33400[i]=-2 and e33600[i]>=0 and e34402[i]>0) then biwkly[i]=e33600[i]/(2*e34402[i]);
   if e34400[i]>=0 and e34402[i]>0 then biwkly[i]=e34400[i]/(2*e34402[i]);
   /*missing value*/
   if -4<e34402[i]<0 then biwkly[i]=e34402[i];

/*received compensation*/
  /*with overtime*/
   if (e33500[i]>=0 and e34428[i]>0) then biwkly[i]=e33500[i]/(2*e34428[i]);
   if (e33500[i]=-2 and e33600[i]>=0 and e34428[i]>0) then biwkly[i]=e33600[i]/(2*e34428[i]);
   if e34400[i]>=0 and e34428[i]>0 then biwkly[i]=e34400[i]/(2*e34428[i]);
   /*missing value*/
   if -4<e34428[i]<0 then biwkly[i]=e34428[i];
   if (e33500[i]>=0 or e33600[i]>=0 or e34400[i]>=0) and e34428[i]=0 then biwkly[i]=-3;

  /*without overtime*/
   if (e33500[i]>=0 and e34402[i]>0) then biwkly[i]=e33500[i]/(2*e34402[i]);
   if (e33500[i]=-2 and e33600[i]>=0 and e34402[i]>0) then biwkly[i]=e33600[i]/(2*e34402[i]);
   if e34400[i]>=0 and e34402[i]>0 then biwkly[i]=e34400[i]/(2*e34402[i]);
   /*missing value*/
   if -4<e34402[i]<0 then biwkly[i]=e34402[i];
   if (e33400[i]>=0 or e33500[i]>=0 or e33600[i]>=0 or e34400[i]>=0) and e34402[i]=0 then biwkly[i]=-3;

/*if still paid hourly...*/
   if e36200[i]>=0 then biwkly[i]=e36200[i];
   if e36100[i]>=0 then biwkly[i]=e36100[i];

end;

 if e83100[i]=4 then do;

/*no compensation*/
   if (e97300[i]>=0 and e98402[i]>0) then biwkly[i]=e97300[i]/(2*e98402[i]);
   if (e97300[i]=-2 and e97500[i]>=0 and e98402[i]>0) then biwkly[i]=e97500[i]/(2*e98402[i]);
   if e98300[i]>=0 and e98402[i]>0 then biwkly[i]=e98300[i]/(2*e98402[i]);
   /*missing value*/
   if -4<e98402[i]<0 then biwkly[i]=e98402[i];

/*received compensation*/
  /*with overtime*/
   if (e97400[i]>=0 and e98429[i]>0) then biwkly[i]=e97400[i]/(2*e98429[i]);
   if (e97400[i]=-2 and e97500[i]>=0 and e98429[i]>0) then biwkly[i]=e97500[i]/(2*e98429[i]);
   if e98300[i]>=0 and e98429[i]>0 then biwkly[i]=e98300[i]/(2*e98429[i]);
   /*missing value*/
   if -4<e98429[i]<0 then biwkly[i]=e98429[i];
```

```
   if (e97400[i]>=0 or e97500[i]>=0 or e98300[i]>=0) and e98429[i]=0 then biwkly[i]=-3;

  /*without overtime*/
   if (e97400[i]>=0 and e98402[i]>0) then biwkly[i]=e97400[i]/(2*e98402[i]);
   if (e97400[i]=-2 and e97500[i]>=0 and e98402[i]>0) then biwkly[i]=e97500[i]/(2*e98402[i]);
   if e98300[i]>=0 and e98402[i]>0 then biwkly[i]=e98300[i]/(2*e98402[i]);
   /*missing value*/
   if -4<e98402[i]<0 then biwkly[i]=e98402[i];
   if (e97300[i]>=0 or e97400[i]>=0 or e97500[i]>=0 or e98300[i]>=0) and e98402[i]=0 then biwkly[i]=-3;

/*if still paid hourly...*/
   if e100100[i]>=0 then biwkly[i]=e100100[i];
   if e100000[i]>=0 then biwkly[i]=e100000[i];

end;
end;

/*set up the hourly wage for youths who report their wage in monthly units.*/
month01=-4;     month02=-4;     month03=-4;     month04=-4;     month05=-4;
month06=-4;     month07=-4;     month08=-4;     month09=-4;

array month month01 month02 month03 month04 month05 month06 month07 month08 month09;

do i=1 to 9;

/* month start wage divided by the number of hours worked per week*/

if e19200[i]=5 then do;

/*no compensation*/
   if (e33400[i]>=0 and e34402[i]>0) then month[i]=e33400[i]/(4.3*e34402[i]);
   if (e33400[i]=-2 and e33600[i]>=0 and e34402[i]>0) then month[i]=e33600[i]/(4.3*e34402[i]);
   if e34400[i]>=0 and e34402[i]>0 then month[i]=e34400[i]/(4.3*e34402[i]);
   /*missing value*/
   if -4<e34402[i]<0 then month[i]=e34402[i];

/*received compensation*/
  /*with overtime*/
   if (e33500[i]>=0 and e34428[i]>0) then month[i]=e33500[i]/(4.3*e34428[i]);
   if (e33500[i]=-2 and e33600[i]>=0 and e34428[i]>0) then month[i]=e33600[i]/(4.3*e34428[i]);
   if e34400[i]>=0 and e34428[i]>0 then month[i]=e34400[i]/(4.3*e34428[i]);
   /*missing value*/
   if -4<e34428[i]<0 then month[i]=e34428[i];
   if (e33500[i]>=0 or e33600[i]>=0 or e34400[i]>=0) and e34428[i]=0 then month[i]=-3;

  /*without overtime*/
   if (e33500[i]>=0 and e34402[i]>0) then month[i]=e33500[i]/(4.3*e34402[i]);
   if (e33500[i]=-2 and e33600[i]>=0 and e34402[i]>0) then month[i]=e33600[i]/(4.3*e34402[i]);
   if e34400[i]>=0 and e34402[i]>0 then month[i]=e34400[i]/(4.3*e34402[i]);
   /*missing value*/
   if -4<e34402[i]<0 then month[i]=e34402[i];
   if (e33400[i]>=0 or e33500[i]>=0 or e33600[i]>=0 or e34400[i]>=0) and e34402[i]=0 then month[i]=-3;

/*if still paid hourly...*/
   if e36200[i]>=0 then month[i]=e36200[i];
   if e36100[i]>=0 then month[i]=e36100[i];
```

```
end;

 if e83100[i]=5 then do;

/*no compensation*/
   if (e97300[i]>=0 and e98402[i]>0) then month[i]=e97300[i]/(4.3*e98402[i]);
   if (e97300[i]=-2 and e97500[i]>=0 and e98402[i]>0) then month[i]=e97500[i]/(4.3*e98402[i]);
   if e98300[i]>=0 and e98402[i]>0 then month[i]=e98300[i]/(4.3*e98402[i]);
   /*missing value*/
   if -4<e98402[i]<0 then month[i]=e98402[i];

/*received compensation*/
  /*with overtime*/
   if (e97400[i]>=0 and e98429[i]>0) then month[i]=e97400[i]/(4.3*e98429[i]);
   if (e97400[i]=-2 and e97500[i]>=0 and e98429[i]>0) then month[i]=e97500[i]/(4.3*e98429[i]);
   if e98300[i]>=0 and e98429[i]>0 then month[i]=e98300[i]/(4.3*e98429[i]);
   /*missing value*/
   if -4<e98429[i]<0 then month[i]=e98429[i];
   if (e97400[i]>=0 or e97500[i]>=0 or e98300[i]>=0) and e98429[i]=0 then month[i]=-3;

  /*without overtime*/
   if (e97400[i]>=0 and e98402[i]>0) then month[i]=e97400[i]/(4.3*e98402[i]);
   if (e97400[i]=-2 and e97500[i]>=0 and e98402[i]>0) then month[i]=e97500[i]/(4.3*e98402[i]);
   if e98300[i]>=0 and e98402[i]>0 then month[i]=e98300[i]/(4.3*e98402[i]);
   /*missing value*/
   if -4<e98402[i]<0 then month[i]=e98402[i];
   if (e97300[i]>=0 or e97400[i]>=0 or e97500[i]>=0 or e98300[i]>=0) and e98402[i]=0 then month[i]=-3;

/*if still paid hourly...*/
   if e100100[i]>=0 then month[i]=e100100[i];
   if e100000[i]>=0 then month[i]=e100000[i];

end;
end;

/*set up the hourly wage for youths who report their wage in annual units.*/
annual01=-4;      annual02=-4;      annual03=-4;      annual04=-4;      annual05=-4;
annual06=-4;      annual07=-4;      annual08=-4;      annual09=-4;

array annual annual01 annual02 annual03 annual04 annual05 annual06 annual07 annual08 annual09;

do i=1 to 9;

/* annual start wage divided by the number of hours worked per week*/

if e19200[i]=6 then do;

/*no compensation*/
   if (e33400[i]>=0 and e34402[i]>0 and e35600[i]>0) then annual[i]=e33400[i]/(e35600[i]*e34402[i]);
   if (e33400[i]=-2 and e33600[i]>=0 and e34402[i]>0 and e35600[i]>0) then
            annual[i]=e33600[i]/(e35600[i]*e34402[i]);
   if e34400[i]>=0 and e34402[i]>0 and e35600[i]>0 then annual[i]=e34400[i]/(e35600[i]*e34402[i]);
   /*missing value*/
   if -4<e34402[i]<0 then annual[i]=e34402[i];

/*received compensation*/
  /*with overtime*/
```

```
  if (e33500[i]>=0 and e34428[i]>0 and e35600[i]>0) then annual[i]=e33500[i]/(e35600[i]*e34428[i]);
  if (e33500[i]=-2 and e33600[i]>=0 and e34428[i]>0 and e35600[i]>0) then
          annual[i]=e33600[i]/(e35600[i]*e34428[i]);
  if e34400[i]>=0 and e34428[i]>0 and e35600[i]>0 then annual[i]=e34400[i]/(e35600[i]*e34428[i]);
  /*missing value*/
  if -4<e34428[i]<0 then annual[i]=e34428[i];
  if (e33500[i]>=0 or e33600[i]>=0 or e34400[i]>=0) and e34428[i]=0 then annual[i]=-3;

 /*without overtime*/
  if (e33500[i]>=0 and e34402[i]>0 and e35600[i]>0) then annual[i]=e33500[i]/(e35600[i]*e34402[i]);
  if (e33500[i]=-2 and e33600[i]>=0 and e34402[i]>0 and e35600[i]>0) then
          annual[i]=e33600[i]/(e35600[i]*e34402[i]);
  if e34400[i]>=0 and e34402[i]>0 and e35600[i]>0 then annual[i]=e34400[i]/(e35600[i]*e34402[i]);
  /*missing value*/
  if -4<e34402[i]<0 then annual[i]=e34402[i];
  if (e33400[i]>=0 or e33500[i]>=0 or e33600[i]>=0 or e34400[i]>=0) and e34402[i]=0 then annual[i]=-3;
  if -4<e35600[i]<0 then annual[i]=e35600[i];
  if e35600[i]=0 then annual[i]=-3;

/*if still paid hourly...*/
  if e36200[i]>=0 then annual[i]=e36200[i];
  if e36100[i]>=0 then annual[i]=e36100[i];

end;

if e83100[i]=6 then do;

/*no compensation*/
  if (e97300[i]>=0 and e98402[i]>0 and e99500[i]>0) then annual[i]=e97300[i]/(e99500[i]*e98402[i]);
  if (e97300[i]=-2 and e97500[i]>=0 and e98402[i]>0 and e99500[i]>0) then
          annual[i]=e97500[i]/(e99500[i]*e98402[i]);
  if e98300[i]>=0 and e98402[i]>0 and e99500[i]>0 then annual[i]=e98300[i]/(e99500[i]*e98402[i]);
  /*missing value*/
  if -4<e98402[i]<0 then annual[i]=e98402[i];

/*received compensation*/
 /*with overtime*/
  if (e97400[i]>=0 and e98429[i]>0 and e99500[i]>0) then annual[i]=e97400[i]/(e99500[i]*e98429[i]);
  if (e97400[i]=-2 and e97500[i]>=0 and e98429[i]>0 and e99500[i]>0) then
          annual[i]=e97500[i]/(e99500[i]*e98429[i]);
  if e98300[i]>=0 and e98429[i]>0 and e99500[i]>0 then annual[i]=e98300[i]/(e99500[i]*e98429[i]);
  /*missing value*/
  if -4<e99500[i]<0 then annual[i]=e99500[i];
  if -4<e98429[i]<0 then annual[i]=e98429[i];
  if (e97400[i]>=0 or e97500[i]>=0 or e98300[i]>=0) and e98429[i]=0 then annual[i]=-3;

 /* without overtime*/
  if (e97400[i]>=0 and e98402[i]>0 and e99500[i]>0) then annual[i]=e97400[i]/(e99500[i]*e98402[i]);
  if (e97400[i]=-2 and e97500[i]>=0 and e98402[i]>0 and e99500[i]>0) then
          annual[i]=e97500[i]/(e99500[i]*e98402[i]);
  if e98300[i]>=0 and e98402[i]>0 and e99500[i]>0 then annual[i]=e98300[i]/(e99500[i]*e98402[i]);
  /*missing value*/
  if -4<e99500[i]<0 then annual[i]=e99500[i];
  if -4<e98402[i]<0 then annual[i]=e98402[i];
  if (e97300[i]>=0 or e97400[i]>=0 or e97500[i]>=0 or e98300[i]>=0) and e98402[i]=0 then annual[i]=-3;
  if -4<e99500[i]<0 then annual[i]=e99500[i];
  if e99500[i]=0 then annual[i]=-3;
```

```
/*if still paid hourly...*/
   if e100100[i]>=0 then annual[i]=e100100[i];
   if e100000[i]>=0 then annual[i]=e100000[i];

end;
end;

/*set up the hourly wage for youths who report their wage in semim units.*/
semim01=-4;      semim02=-4;      semim03=-4;      semim04=-4;      semim05=-4;
semim06=-4;      semim07=-4;      semim08=-4;      semim09=-4;

array semim semim01 semim02 semim03 semim04 semim05 semim06 semim07 semim08 semim09;

do i=1 to 9;

/* semim start wage divided by the number of hours worked per week*/

if e19200[i]=8 then do;

/*no compensation*/
   if (e33400[i]>=0 and e34402[i]>0) then semim[i]=e33400[i]/(2.15*e34402[i]);
   if (e33400[i]=-2 and e33600[i]>=0 and e34402[i]>0) then semim[i]=e33600[i]/(2.15*e34402[i]);
   if e34400[i]>=0 and e34402[i]>0 then semim[i]=e34400[i]/(2.15*e34402[i]);
   /*missing value*/
   if -4<e34402[i]<0 then semim[i]=e34402[i];

/*received compensation*/
  /*with overtime*/
   if (e33500[i]>=0 and e34428[i]>0) then semim[i]=e33500[i]/(2.15*e34428[i]);
   if (e33500[i]=-2 and e33600[i]>=0 and e34428[i]>0) then semim[i]=e33600[i]/(2.15*e34428[i]);
   if e34400[i]>=0 and e34428[i]>0 then semim[i]=e34400[i]/(2.15*e34428[i]);
   /*missing value*/
   if -4<e34428[i]<0 then semim[i]=e34428[i];
   if (e33500[i]>=0 or e33600[i]>=0 or e34400[i]>=0) and e34428[i]=0 then semim[i]=-3;

  /*without overtime*/
   if (e33500[i]>=0 and e34402[i]>0) then semim[i]=e33500[i]/(2.15*e34402[i]);
   if (e33500[i]=-2 and e33600[i]>=0 and e34402[i]>0) then semim[i]=e33600[i]/(2.15*e34402[i]);
   if e34400[i]>=0 and e34402[i]>0 then semim[i]=e34400[i]/(2.15*e34402[i]);
   /*missing value*/
   if -4<e34402[i]<0 then semim[i]=e34402[i];
   if (e33400[i]>=0 or e33500[i]>=0 or e33600[i]>=0 or e34400[i]>=0) and e34402[i]=0 then semim[i]=-3;

/*if still paid hourly...*/
   if e36200[i]>=0 then semim[i]=e36200[i];
   if e36100[i]>=0 then semim[i]=e36100[i];

end;

if e83100[i]=8 then do;

/*no compensation*/
   if (e97300[i]>=0 and e98402[i]>0) then semim[i]=e97300[i]/(2.15*e98402[i]);
   if (e97300[i]=-2 and e97500[i]>=0 and e98402[i]>0) then semim[i]=e97500[i]/(2.15*e98402[i]);
   if e98300[i]>=0 and e98402[i]>0 then semim[i]=e98300[i]/(2.15*e98402[i]);
   /*missing value*/
```

```
   if -4<e98402[i]<0 then semim[i]=e98402[i];

/*received compensation*/
  /*with overtime*/
   if (e97400[i]>=0 and e98429[i]>0) then semim[i]=e97400[i]/(2.15*e98429[i]);
   if (e97400[i]=-2 and e97500[i]>=0 and e98429[i]>0) then semim[i]=e97500[i]/(2.15*e98429[i]);
   if e98300[i]>=0 and e98429[i]>0 then semim[i]=e98300[i]/(2.15*e98429[i]);
   /*missing value*/
   if -4<e98429[i]<0 then semim[i]=e98429[i];
   if (e97400[i]>=0 or e97500[i]>=0 or e98300[i]>=0) and e98429[i]=0 then semim[i]=-3;

  /*without overtime*/
   if (e97400[i]>=0 and e98402[i]>0) then semim[i]=e97400[i]/(2.15*e98402[i]);
   if (e97400[i]=-2 and e97500[i]>=0 and e98402[i]>0) then semim[i]=e97500[i]/(2.15*e98402[i]);
   if e98300[i]>=0 and e98402[i]>0 then semim[i]=e98300[i]/(2.15*e98402[i]);
   /*missing value*/
   if -4<e98402[i]<0 then semim[i]=e98402[i];
   if (e97300[i]>=0 or e97400[i]>=0 or e97500[i]>=0 or e98300[i]>=0) and e98402[i]=0 then semim[i]=-3;

/*if still paid hourly...*/
   if e100100[i]>=0 then semim[i]=e100100[i];
   if e100000[i]>=0 then semim[i]=e100000[i];

end;
end;

/*set up the hourly wage for youths who report their wage by item or job.*/
item01=-4;      item02=-4;      item03=-4;      item04=-4;      item05=-4;
item06=-4;      item07=-4;      item08=-4;      item09=-4;

array item item01 item02 item03 item04 item05 item06 item07 item08 item09;

flag34428=0; flag98429=0;

do i=1 to 9;

 if e19200[i] in (12,13) then do;

   /* first, get the total pay for each item or job, we name this variable as pay. */
   if e33400[i]>=0 then pay[i]=e33400[i]; /*no comp. */
   if e33500[i]>=0 then pay[i]=e33500[i]; /*with comp. */
   if e33600[i]>=0 then pay[i]=e33600[i]; /*if -2 for one of e33400 or e33500. */
   if e34400[i]>=0 then pay[i]=e34400[i]; /*if previous entry is incorrect. */

   /* without overtime, where e34402 will be the weekly hour info. */
      /* condition based on the time unit to finish per item. */
        if e34402aab[i]=1 and e34402aad[i]>0 then item[i]=(pay[i]/e34402aad[i])*60;
        if e34402aab[i]=2 and e34402aad[i]>0 then item[i]=pay[i]/e34402aad[i];
        if e34402aab[i]=3 and e34402aad[i]>0 and e34402ab[i]>=0 and e34402[i]>0 then
          item[i]=(pay[i]/e34402aad[i])*e34402ab[i]/e34402[i];
        if e34402aab[i]=4 and e34402aad[i]>0 and e34402[i]>0 then item[i]=(pay[i]/e34402aad[i])/e34402[i];
        if e34402aab[i]=5 and e34402aad[i]>0 and e34402[i]>0 then item[i]=(pay[i]/e34402aad[i])/(e34402[i]*4.3);
      /* missing value */
        if -4<e34402aab[i]<0 or -4<e34402aad[i]<=0 then item[i]=-3;
        if e34402aab[i] in (3,4,5) and -4<e34402[i]<=0 then item[i]=-3;
        if e34402aab[i]=3 and e34402ab[i]<0 then item[i]=-3;
```

```
   /* with overtime, where e34428 will be the weekly hour info. and there is no info. available to calculate
   hourly pay. */

   if e212001[i]=1 then do;
      item[i]=-3;
      flag34428=1;
   end;

/*if still paid hourly...*/
   if e36200[i]>=0 then item[i]=e36200[i];
   if e36100[i]>=0 then item[i]=e36100[i];

end;

if e83100[i] in (12,13) then do;

   /* first, get the total pay for each item or job, we name this variable as pay. */
   if e97300[i]>=0 then pay[i]=e97300[i]; /*no comp. */
   if e97400[i]>=0 then pay[i]=e97400[i]; /*with comp. */
   if e97500[i]>=0 then pay[i]=e97500[i]; /*if -2 for one of e33400 or e33500. */
   if e98300[i]>=0 then pay[i]=e98300[i]; /*if previous entry is incorrect. */

   /* without overtime, where e98402 will be the weekly hour info. */
      /* condition based on the time unit to finish per item. */
        if e98402caa[i]=1 and e98402cab[i]>0 then item[i]=(pay[i]/e98402cab[i])*60;
        if e98402caa[i]=2 and e98402cab[i]>0 then item[i]=pay[i]/e98402cab[i];
        if e98402caa[i]=3 and e98402cab[i]>0 and e98402d[i]>=0 and e98402[i]>0 then
          item[i]=(pay[i]/e98402cab[i])*e98402d[i]/e98402[i];
        if e98402caa[i]=4 and e98402cab[i]>0 and e98402[i]>0 then item[i]=(pay[i]/e98402cab[i])/e98402[i];
        if e98402caa[i]=5 and e98402cab[i]>0 and e98402[i]>0 then item[i]=(pay[i]/e98402cab[i])/(e98402[i]*4.3);
      /* missing value */
        if -4<e98402caa[i]<0 or -4<e98402cab[i]<=0 then item[i]=-3;
        if e98402caa[i] in (3,4,5) and -4<e98402[i]<=0 then item[i]=-3;
        if e98402caa[i]=3 and e98402d[i]<0 then item[i]=-3;

   /* with overtime, where e98429 will be the weekly hour info. and there is no info. available to calculate
   hourly pay. */

   if e83100[i]=1 then do; item[i]=-3; flag98429=1; end;

/*if still paid hourly...*/
   if e100100[i]>=0 then item[i]=e100100[i];
   if e100000[i]>=0 then item[i]=e100000[i];

end;
end;

/*for the people who report other time units. */
array otherf otherf1-otherf9;

do i=1 to 9; otherf[i]=0; end;

do i=1 to 9;

if e19200[i] in (7,9,12,13,15,16,17,14,18,19,21,22,23,24,25,26,28) or e83100[i] in
          (7,9,12,13,15,16,17,14,18,19,21,22,23,24,25,26,28) then otherf[i]=otherf[i]+1;
end;
```

```
/******** create the hourly rate of pay based on the start wage **********************/
hrwg01=-4;      hrwg02=-4;      hrwg03=-4;      hrwg04=-4;      hrwg05=-4;
hrwg06=-4;      hrwg07=-4;      hrwg08=-4;      hrwg09=-4;

array hrwg hrwg01 hrwg02 hrwg03 hrwg04 hrwg05 hrwg06 hrwg07 hrwg08 hrwg09;

/* report hourly wage to be 0 for the family business and to be -3 for the poorly reported time units.*/
do i=1 to 9;
   if e19200[i] in (9,14) or e83100[i] in (9,14)  then hrwg[i]=0;
end;

/* report hourly rate -1 or -2 if amount is -1 or -2*/
do i=1 to 9;
  if e19200[i] in (2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) then do;
   if e33400[i]=-1  or e33500[i]=-1 then hrwg[i]=-1;
   if (e33400[i]=-2 and e33600[i]=-2) or (e33500[i]=-2 and e33600[i]=-2) then hrwg[i]=-2;
   if e33400[i]=-3 or e33500[i]=-3 then hrwg[i]=-3;
  end;

  if e83100[i] in (2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) then do;
   if e97300[i]=-1 or e97400[i]=-1 then hrwg[i]=-1;
   if (e97300[i]=-2 and e97500[i]=-2) or (e97400[i]=-2 and e97500[i]=-2) then hrwg[i]=-2;
   if e97300[i]=-3 or e97400[i]=-3 then hrwg[i]=-3;
  end;
end;

/* report hourly rate -3 if no hours reported*/
do i=1 to 9;
   if e212001[i]=1 and e34428[i]=-4 then hrwg[i]=-3;
end;

do i=1 to 9;
if annual[i] ge 0 then hrwg[i]=annual[i];        if month[i] ge 0 then hrwg[i]=month[i];
if biwkly[i] ge 0 then hrwg[i]=biwkly[i];        if weekly[i] ge 0 then hrwg[i]=weekly[i];
if daily[i] ge 0 then hrwg[i]=daily[i];          if hrwage[i] ge 0 then hrwg[i]=hrwage[i];
if semim[i] ge 0 then hrwg[i]=semim[i];          if item[i] ge 0 then hrwg[i]=item[i];
if hrwage[i] eq -1 or daily[i]=-1 or weekly[i] eq -1 or biwkly[i] eq -1 or month[i] eq -1 or annual[i] eq -1 or semim[i]=-1
          or item[i]=-1 then hrwg[i]=-1;
if hrwage[i] eq -2 or daily[i]=-2 or weekly[i] eq -2 or biwkly[i] eq -2 or month[i] eq -2 or annual[i] eq -2 or semim[i]=-2
          or item[i]=-2 then hrwg[i]=-2;
if hrwage[i] eq -3 or daily[i]=-3 or weekly[i] eq -3 or biwkly[i] eq -3 or month[i] eq -3 or annual[i] eq -3 or semim[i]=-3
          or item[i]=-3 then hrwg[i]=-3;

end;

/* report the corrected wage if the correction is made */

do i=1 to 9;
  if e226041[i]=1 and e226042[i]=0 then do; /* rate incorrect but hours correct*/

    /* no overtime, use e34402 for hours */
    if e22609[i]=1 and e22626[i]>=0 then hrwg[i]=e22626[i];
    if e22609[i]=2 and e34402[i]>0 and e22626e[i]>0 and e22626[i]>=0 then hrwg[i]=e22626[i]*e22626e[i]/e34402[i];
    if e22609[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28, 999,-1,-2) and e34402[i]>0 and e22626[i]>=0 then
          hrwg[i]=e22626[i]/e34402[i];
    if e22609[i]=4 and e34402[i]>0 and e22626[i]>=0 then hrwg[i]=e22626[i]/(2*e34402[i]);
```

```
  if e22609[i]=5 and e34402[i]>0 and e22626[i]>=0 then hrwg[i]=e22626[i]/(4.3*e34402[i]);
  if e22609[i]=6 and e34402[i]>0 and e35600[i]>0 and e22626[i]>=0 then hrwg[i]=e22626[i]/(e35600[i]*e34402[i]);
  if e22609[i]=8 and e34402[i]>0 and e22626[i]>=0 then hrwg[i]=e22626[i]/(2.15*e34402[i]);
  if e22609[i] in (9,14) then hrwg[i]=0;
  if e22609[i]=2 and e22626e[i] le 0 then hrwg[i]=-3;
  if e22609[i]=2 and -4<e22626e[i]<0 then hrwg[i]=e22626e[i];
  if e22609[i]=6 and e35600[i] le 0 then hrwg[i]=-3;
  if e22609[i]=6 and -4<e35600[i]<0 then hrwg[i]=e35600[i];
  if e22609[i] in (12,13) and e22626[i]>=0 then do;
    /* condition based on the time unit to finish per hrwg. */
     if e22626c[i]=1 and e22626cb[i]>0 then hrwg[i]=(e22626[i]/e22626cb[i])*60;
     if e22626c[i]=2 and e22626cb[i]>0 then hrwg[i]=e22626[i]/e22626cb[i];
     if e22626c[i]=3 and e22626cb[i]>0 and e22626e[i]>=0 and e34402[i]>0 then
       hrwg[i]=(e22626[i]/e22626cb[i])*e22626e[i]/e34402[i];
     if e22626c[i]=4 and e22626cb[i]>0 and e34402[i]>0 then hrwg[i]=(e22626[i]/e22626cb[i])/e34402[i];
     if e22626c[i]=5 and e22626cb[i]>0 and e34402[i]>0 then hrwg[i]=(e22626[i]/e22626cb[i])/(e34402[i]*4.3);
   /* missing value */
     if -4<e22626c[i]<0 or -4<e22626cb[i]<=0 then hrwg[i]=-3;
     if e22626c[i] in (3,4,5) and -4<e34402[i]<=0 then hrwg[i]=-3;
     if e22626c[i]=3 and e22626e[i]<0 then hrwg[i]=-3;
  end;

  if e22609[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e22626[i]<0 then
          hrwg[i]=e22626[i];
  if e22609[i] in (2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then hrwg[i]=-
          3;
  if e22609[i] in (2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          hrwg[i]=e34402[i];

  /* overtime, use e34428 for hours*/
  if e22609[i]=1 and e22626[i]>=0 then hrwg[i]=e22626[i];
  if e22609[i]=2 and e34428[i]>0 and e22626e[i]>0 and e22626[i]>=0 then hrwg[i]=e22626[i]*e22626e[i]/e34428[i];
  if e22609[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28, -1,-2,999) and e34428[i]>0 and e22626[i]>=0 then
          hrwg[i]=e22626[i]/e34428[i];
  if e22609[i]=4 and e34428[i]>0 and e22626[i]>=0 then hrwg[i]=e22626[i]/(2*e34428[i]);
  if e22609[i]=5 and e34428[i]>0 and e22626[i]>=0 then hrwg[i]=e22626[i]/(4.3*e34428[i]);
  if e22609[i]=6 and e34428[i]>0 and e35600[i]>0 and e22626[i]>=0 then hrwg[i]=e22626[i]/(e35600[i]*e34428[i]);
  if e22609[i]=8 and e34428[i]>0 and e22626[i]>=0 then hrwg[i]=e22626[i]/(2.15*e34428[i]);
  if e22609[i] in (12,13) and e22626[i]>=0 then do;
    /* condition based on the time unit to finish per hrwg. */
     if e22626c[i]=3 and e22626cb[i]>0 and e22626e[i]>=0 and e34428[i]>0 then
       hrwg[i]=(e22626[i]/e22626cb[i])*e22626e[i]/e34428[i];
     if e22626c[i]=4 and e22626cb[i]>0 and e34428[i]>0 then hrwg[i]=(e22626[i]/e22626cb[i])/e34428[i];
     if e22626c[i]=5 and e22626cb[i]>0 and e34428[i]>0 then hrwg[i]=(e22626[i]/e22626cb[i])/(e34428[i]*4.3);
   /* missing value */
     if e22626c[i] in (3,4,5) and -4<e34428[i]<=0 then hrwg[i]=-3;
  end;

  if e22609[i] in (2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then hrwg[i]=-
          3;
  if e22609[i] in (2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
          hrwg[i]=e34428[i];
end;

if e226041[i]=0 and e226042[i]=1 then do;  /*rate correct but hours incorrect*/
  if e19200[i] ne 1 and e22610[i]>0 and e34402[i]>0 and hrwg[i]>=0 then hrwg[i]=hrwg[i]*e34402[i]/e22610[i];
  if e19200[i] ne 1 and e22610[i]>0 and e34428[i]>0 and hrwg[i]>=0 then hrwg[i]=hrwg[i]*e34428[i]/e22610[i];
```

```
    if e19200[i] ne 1 and e22610[i]=0 then hrwg[i]=-3;
    if e19200[i] ne 1 and -4<e22610[i]<0 then hrwg[i]=e22610[i];
  end;

  if e226041[i]=1 and e226042[i]=1 then do; /* neither rate nor hours is correct*/
    if e22609[i]=1 and e22626[i]>=0 then hrwg[i]=e22626[i];
    if e22609[i]=2 and e22610[i]>0 and e22626e[i]>0 and e22626[i]>=0 then hrwg[i]=e22626[i]*e22626e[i]/e22610[i];
    if e22609[i]=3 and e22610[i]>0 and e22626[i]>=0 then hrwg[i]=e22626[i]/e22610[i];
    if e22609[i]=4 and e22610[i]>0 and e22626[i]>=0 then hrwg[i]=e22626[i]/(2*e22610[i]);
    if e22609[i]=5 and e22610[i]>0 and e22626[i]>=0 then hrwg[i]=e22626[i]/(4.3*e22610[i]);
    if e22609[i]=6 and e22610[i]>0 and e35600[i]>0 and e22626[i]>=0 then hrwg[i]=e22626[i]/(e35600[i]*e22610[i]);
    if e22609[i]=8 and e22610[i]>0 and e22626[i]>=0 then hrwg[i]=e22626[i]/(2.15*e22610[i]);
    if e22609[i] in (9,14) then hrwg[i]=0;
    if e22609[i]=2 and e22626e[i] le 0 then hrwg[i]=-3;
    if e22609[i]=6 and e35600[i] le 0 then hrwg[i]=-3;
    if e22609[i] in (12,13) and e22626[i]>=0 then do;
      /* condition based on the time unit to finish per hrwg. */
       if e22626c[i]=1 and e22626cb[i]>0 then hrwg[i]=(e22626[i]/e22626cb[i])*60;
       if e22626c[i]=2 and e22626cb[i]>0 then hrwg[i]=e22626[i]/e22626cb[i];
       if e22626c[i]=3 and e22626cb[i]>0 and e22626e[i]>=0 and e22610[i]>0 then
         hrwg[i]=(e22626[i]/e22626cb[i])*e22626e[i]/e22610[i];
       if e22626c[i]=4 and e22626cb[i]>0 and e22610[i]>0 then hrwg[i]=(e22626[i]/e22626cb[i])/e22610[i];
       if e22626c[i]=5 and e22626cb[i]>0 and e22610[i]>0 then hrwg[i]=(e22626[i]/e22626cb[i])/(e22610[i]*4.3);
     /* missing value */
       if -4<e22626c[i]<0 or -4<e22626cb[i]<=0 then hrwg[i]=-3;
       if e22626c[i] in (3,4,5) and -4<e22610[i]<=0 then hrwg[i]=-3;
       if e22626c[i]=3 and e22626e[i]<0 then hrwg[i]=-3;
    end;
    if e22609[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e22626[i]<0 then
            hrwg[i]=e22626[i];
    if e22609[i] in (2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e22610[i]=0 then hrwg[i]=-
            3;
    if e22609[i] in (2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e22610[i]<0 then
            hrwg[i]=e22610[i];
  end;
end;

do i=1 to 9;
  if e102251[i]=1 and e102252[i]=0 then do; /* rate incorrect but hours correct*/

    /* no overtime, use e98402 for hours */
    if e100230[i]=1 and e100248[i]>=0 then hrwg[i]=e100248[i];
    if e100230[i]=2 and e98402[i]>0 and e100248e[i]>0 and e100248[i]>=0 then
            hrwg[i]=e100248[i]*e100248e[i]/e98402[i];
    if e100230[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e98402[i]>0 and e100248[i]>=0
            then hrwg[i]=e100248[i]/e98402[i];
    if e100230[i]=4 and e98402[i]>0 and e100248[i]>=0 then hrwg[i]=e100248[i]/(2*e98402[i]);
    if e100230[i]=5 and e98402[i]>0 and e100248[i]>=0 then hrwg[i]=e100248[i]/(4.3*e98402[i]);
    if e100230[i]=6 and e98402[i]>0 and e99500[i]>0 and e100248[i]>=0 then
            hrwg[i]=e100248[i]/(e99500[i]*e98402[i]);
    if e100230[i]=8 and e98402[i]>0 and e100248[i]>=0 then hrwg[i]=e100248[i]/(2.15*e98402[i]);
    if e100230[i] in (9,14) then hrwg[i]=0;
    if e100230[i]=2 and e100248e[i] le 0 then hrwg[i]=-3;
    if e100230[i]=2 and -4<e100248e[i]<0 then hrwg[i]=e98402d[i];
    if e100230[i]=6 and e99500[i] le 0 then hrwg[i]=-3;
    if e100230[i]=6 and -4<e99500[i]<0 then hrwg[i]=e99500[i];
    if e100230[i] in (12,13) and e100248[i]>=0 then do;
```

```
    /* condition based on the time unit to finish per hrwg. */
     if e100248c[i]=1 and e100248cb[i]>0 then hrwg[i]=(e100248[i]/e100248cb[i])*60;
     if e100248c[i]=2 and e100248cb[i]>0 then hrwg[i]=e100248[i]/e100248cb[i];
     if e100248c[i]=3 and e100248cb[i]>0 and e100248e[i]>=0 and e98402[i]>0 then
       hrwg[i]=(e100248[i]/e100248cb[i])*e100248e[i]/e98402[i];
     if e100248c[i]=4 and e100248cb[i]>0 and e98402[i]>0 then hrwg[i]=(e100248[i]/e100248cb[i])/e98402[i];
     if e100248c[i]=5 and e100248cb[i]>0 and e98402[i]>0 then hrwg[i]=(e100248[i]/e100248cb[i])/(e98402[i]*4.3);
   /* missing value */
     if -4<e100248c[i]<0 or -4<e100248cb[i]<=0 then hrwg[i]=-3;
     if e100248c[i] in (3,4,5) and -4<e98402[i]<=0 then hrwg[i]=-3;
     if e100248c[i]=3 and e100248e[i]<0 then hrwg[i]=-3;
  end;

  if e100230[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e100248[i]<0 then
          hrwg[i]=e100248[i];
  if e100230[i] in (2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98402[i]=0 then hrwg[i]=-
          3;
  if e100230[i] in (2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98402[i]<0 then
          hrwg[i]=e98402[i];

  /* overtime, use e98429 for hours*/
  if e100230[i]=1 and e100248[i]>=0 then hrwg[i]=e100248[i];
  if e100230[i]=2 and e98429[i]>0 and e100248e[i]>0 and e100248[i]>=0 then
          hrwg[i]=e100248[i]*e100248e[i]/e98429[i];
  if e100230[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28, 999,-1,-2) and e98429[i]>0 and e100248[i]>=0
          then hrwg[i]=e100248[i]/e98429[i];
  if e100230[i]=4 and e98429[i]>0 and e100248[i]>=0 then hrwg[i]=e100248[i]/(2*e98429[i]);
  if e100230[i]=5 and e98429[i]>0 and e100248[i]>=0 then hrwg[i]=e100248[i]/(4.3*e98429[i]);
  if e100230[i]=6 and e98429[i]>0 and e99500[i]>0 and e100248[i]>=0 then
          hrwg[i]=e100248[i]/(e99500[i]*e98429[i]);
  if e100230[i]=8 and e98429[i]>0 and e100248[i]>=0 then hrwg[i]=e100248[i]/(2.15*e98429[i]);
  if e100230[i] in (12,13) and e100248[i]>=0 then do;
    /* condition based on the time unit to finish per hrwg. */
     if e100248c[i]=3 and e100248cb[i]>0 and e100248e[i]>=0 and e98429[i]>0 then
       hrwg[i]=(e100248[i]/e100248cb[i])*e100248e[i]/e98429[i];
     if e100248c[i]=4 and e100248cb[i]>0 and e98429[i]>0 then hrwg[i]=(e100248[i]/e100248cb[i])/e98429[i];
     if e100248c[i]=5 and e100248cb[i]>0 and e98429[i]>0 then hrwg[i]=(e100248[i]/e100248cb[i])/(e98429[i]*4.3);
   /* missing value */
     if e100248c[i] in (3,4,5) and -4<e98429[i]<=0 then hrwg[i]=-3;
  end;

  if e100230[i] in (2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98429[i]=0 then hrwg[i]=-
          3;
  if e100230[i] in (2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98429[i]<0 then
          hrwg[i]=e98429[i];
end;

if e102251[i]=0 and e102252[i]=1 then do;  /*rate correct but hours incorrect*/
  if e83100[i] ne 1 and e100231[i]>0 and e98402[i]>0 and hrwg[i]>=0 then hrwg[i]=hrwg[i]*e98402[i]/e100231[i];
  if e83100[i] ne 1 and e100231[i]>0 and e98429[i]>0 and hrwg[i]>=0 then hrwg[i]=hrwg[i]*e98429[i]/e100231[i];
  if e83100[i] ne 1 and e100231[i]=0 then hrwg[i]=-3;
  if e83100[i] ne 1 and -4<e100231[i]<0 then hrwg[i]=e100231[i];
end;

if e102251[i]=1 and e102252[i]=1 then do; /* neither rate nor hours is correct*/
  if e100230[i]=1 and e100248[i]>=0 then hrwg[i]=e100248[i];
```

```
   if e100230[i]=2 and e100231[i]>0 and e100248e[i]>0 and e100248[i]>=0 then
          hrwg[i]=e100248[i]*e100248e[i]/e100231[i];
   if e100230[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e100231[i]>0 and e100248[i]>=0
          then hrwg[i]=e100248[i]/e100231[i];
   if e100230[i]=4 and e100231[i]>0 and e100248[i]>=0 then hrwg[i]=e100248[i]/(2*e100231[i]);
   if e100230[i]=5 and e100231[i]>0 and e100248[i]>=0 then hrwg[i]=e100248[i]/(4.3*e100231[i]);
   if e100230[i]=6 and e100231[i]>0 and e99500[i]>0 and e100248[i]>=0 then
          hrwg[i]=e100248[i]/(e99500[i]*e100231[i]);
   if e100230[i]=8 and e100231[i]>0 and e100248[i]>=0 then hrwg[i]=e100248[i]/(2.15*e100231[i]);
   if e100230[i] in (9,14) then hrwg[i]=0;
   if e100230[i]=2 and e100248e[i] le 0 then hrwg[i]=-3;
   if e100230[i]=2 and -4<e100248e[i]<0 then hrwg[i]=e100248e[i];
   if e100230[i]=2 and -4<e100248e[i]<0 then hrwg[i]=e100248e[i];
   if e100230[i]=6 and e99500[i] le 0 then hrwg[i]=-3;
   if e100230[i]=6 and -4<e99500[i]<0 then hrwg[i]=e99500[i];
   if e100230[i] in (12,13) and e100248[i]>=0 then do;
     /* condition based on the time unit to finish per hrwg. */
      if e100248c[i]=1 and e100248cb[i]>0 then hrwg[i]=(e100248[i]/e100248cb[i])*60;
      if e100248c[i]=2 and e100248cb[i]>0 then hrwg[i]=e100248[i]/e100248cb[i];
      if e100248c[i]=3 and e100248cb[i]>0 and e100248e[i]>=0 and e100231[i]>0 then
        hrwg[i]=(e100248[i]/e100248cb[i])*e100248e[i]/e100231[i];
      if e100248c[i]=4 and e100248cb[i]>0 and e100231[i]>0 then hrwg[i]=(e100248[i]/e100248cb[i])/e100231[i];
      if e100248c[i]=5 and e100248cb[i]>0 and e100231[i]>0 then hrwg[i]=(e100248[i]/e100248cb[i])/(e100231[i]*4.3);
    /* missing value */
      if -4<e100248c[i]<0 or -4<e100248cb[i]<=0 then hrwg[i]=-3;
      if e100248c[i] in (3,4,5) and -4<e100231[i]<=0 then hrwg[i]=-3;
      if e100248c[i]=3 and e100248e[i]<0 then hrwg[i]=-3;
   end;
   if e100230[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e100248[i]<0 then
          hrwg[i]=e100248[i];
   if e100230[i] in (2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e100231[i]=0 then
          hrwg[i]=-3;
   if e100230[i] in (2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e100231[i]<0 then
          hrwg[i]=e100231[i];
 end;
end;

/****************** add the start compensation ***********************************************/
/******************************** amount paid of overtime ********************************/
ot1=-4; ot2=-4; ot3=-4; ot4=-4; ot5=-4; ot6=-4; ot7=-4; ot8=-4; ot9=-4;
otf1=0; otf2=0; otf3=0; otf4=0; otf5=0; otf6=0; otf7=0; otf8=0; otf9=0;

multot=0;

array ot ot1-ot9;
array otf otf1-otf9;

/* if report hourly rate of pay */
do i=1 to 9;

if e24502[i]=0 then multot=multot+1;
if e24501[i]>0 and e24502[i]=1 and e24514[i]>=0 then ot[i]=e24514[i];
if e24501[i]>0 and e24502[i]=2 and e24514b[i]>0 and e24514[i]>=0 then ot[i]=e24514[i]*e24514b[i]/e24501[i];
if e24501[i]>0 and e24502[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e24514[i]>=0 then
          ot[i]=e24514[i]/e24501[i];
if e24501[i]>0 and e24502[i]=4 and e24514[i]>=0 then ot[i]=e24514[i]/(2*e24501[i]);
if e24501[i]>0 and e24502[i]=5 and e24514[i]>=0 then ot[i]=e24514[i]/(4.3*e24501[i]);
```

```
if e24501[i]>0 and e24502[i]=6 and e24514[i]>=0 then ot[i]=-3; /* since the weeks per year is unknown.*/
if e24501[i]>0 and e24502[i]=8 and e24514[i]>=0 then ot[i]=e24514[i]/(2.15*e24501[i]);
if e24501[i]>0 and e24502[i]=10 and hrwg[i]>=0 then ot[i]=hrwg[i]*1.5;
if e24501[i]>0 and e24502[i]=11 and hrwg[i]>=0 then ot[i]=hrwg[i]*2;
if e24502[i] in (9, 14) then ot[i]=0;

if -4<e24501[i]<0 then ot[i]=e24501[i];
if e24514[i]>=0 and e24501[i]=0 then ot[i]=-3;

if e24502[i] in (12,13) and e24514[i]>=0 then do;
    /* condition based on the time unit to finish per hrwg. */
      if e24514aab[i]=1 and e24514aabb[i]>0 then ot[i]=(e24514[i]/e24514aabb[i])*60;
      if e24514aab[i]=2 and e24514aabb[i]>0 then ot[i]=e24514[i]/e24514aabb[i];
      if e24514aab[i]=3 and e24514aabb[i]>0 and e24514b[i]>=0 and e24501[i]>0 then
        ot[i]=(e24514[i]/e24514aabb[i])*e24514b[i]/e24501[i];
      if e24514aab[i]=4 and e24514aabb[i]>0 and e24501[i]>0 then ot[i]=(e24514[i]/e24514aabb[i])/e24501[i];
      if e24514aab[i]=5 and e24514aabb[i]>0 and e24501[i]>0 then ot[i]=(e24514[i]/e24514aabb[i])/(e24501[i]*4.3);
     /* missing value */
      if -4<e24514aab[i]<0 or -4<e24514aabb[i]<=0 then ot[i]=-3;
      if e24514aab[i] in (3,4,5) and -4<e24501[i]<=0 then ot[i]=-3;
      if e24514aab[i]=3 and e24514b[i]<0 then ot[i]=-3;
   end;

/*missing value*/;
if -4<e24514[i]<0 then ot[i]=e24514[i];
if e24502[i]=2 and e24514b[i]=0 then ot[i]=-3;
if e24502[i]=2 and -4<e24514b[i]<0 then ot[i]=e24514b[i];

end;

/*if report payment in other units*/

do i=1 to 9;

if e34404[i]=0 then multot=multot+1;
if e34403[i]>0 and e34404[i]=1 and e34413c[i]>=0 then ot[i]=e34413c[i];
if e34403[i]>0 and e34404[i]=2 and e34413e[i]>0 and e34413c[i]>=0 then ot[i]=e34413c[i]*e34413e[i]/e34403[i];
if e34403[i]>0 and e34404[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e34413c[i]>=0 then
          ot[i]=e34413c[i]/e34403[i];
if e34403[i]>0 and e34404[i]=4  and e34413c[i]>=0 then ot[i]=e34413c[i]/(2*e34403[i]);
if e34403[i]>0 and e34404[i]=5  and e34413c[i]>=0 then ot[i]=e34413c[i]/(4.3*e34403[i]);
if e34403[i]>0 and e34404[i]=6  and e34413c[i]>=0 then ot[i]=-3; /* since there is no weeks per year. */
if e34403[i]>0 and e34404[i]=8  and e34413c[i]>=0 then ot[i]=e34413c[i]/(2.15*e34403[i]);
if e34403[i]>0 and e34404[i]=10 and hrwg[i]>=0 then ot[i]=hrwg[i]*1.5;
if e34403[i]>0 and e34404[i]=11 and hrwg[i]>=0 then ot[i]=hrwg[i]*2;
if e34404[i] in (9,14) then ot[i]=0;

if -4<e34403[i]<0 then ot[i]=e34403[i];
if e34413c[i]>=0 and e34403[i]=0 then ot[i]=-3;

if e34404[i] in (12,13) and e34413c[i]>=0 then do;
    /* condition based on the time unit to finish per hrwg. */
      if e34413cc[i]=1 and e34413cd[i]>0 then ot[i]=(e34413c[i]/e34413cd[i])*60;
      if e34413cc[i]=2 and e34413cd[i]>0 then ot[i]=e34413c[i]/e34413cd[i];
      if e34413cc[i]=3 and e34413cd[i]>0 and e34413e[i]>=0 and e34403[i]>0 then
        ot[i]=(e34413c[i]/e34413cd[i])*e34413e[i]/e34403[i];
      if e34413cc[i]=4 and e34413cd[i]>0 and e34403[i]>0 then ot[i]=(e34413c[i]/e34413cd[i])/e34403[i];
```

```
      if e34413cc[i]=5 and e34413cd[i]>0 and e34403[i]>0 then ot[i]=(e34413c[i]/e34413cd[i])/(e34403[i]*4.3);
    /* missing value */
      if -4<e34413cc[i]<0 or -4<e34413cd[i]<=0 then ot[i]=-3;
      if e34413cc[i] in (3,4,5) and -4<e34403[i]<=0 then ot[i]=-3;
      if e34413cc[i]=3 and e34413e[i]<0 then ot[i]=-3;
   end;

/*missing value*/;
if -4<e34413c[i]<0 then ot[i]=e34413c[i];
if e34404[i]=2 and e34413e[i]=0 then ot[i]=-3;
if e34404[i]=2 and -4<e34413e[i]<0 then ot[i]=e34413e[i];

end;

/*if report overtime pay later*/

do i=1 to 9;

if e88502[i]=0 then multot=multot+1;
if e88501[i]>0 and e88502[i]=1  and e88512[i]>=0 then ot[i]=e88512[i];
if e88501[i]>0 and e88502[i]=2  and e88512b[i]>0 and e88512[i]>=0 then ot[i]=e88512[i]*e88512b[i]/e88501[i];
if e88501[i]>0 and e88502[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2)  and e88512[i]>=0 then
          ot[i]=e88512[i]/e88501[i];
if e88501[i]>0 and e88502[i]=4  and e88512[i]>=0 then ot[i]=e88512[i]/(2*e88501[i]);
if e88501[i]>0 and e88502[i]=5  and e88512[i]>=0 then ot[i]=e88512[i]/(4.3*e88501[i]);
if e88501[i]>0 and e88502[i]=6  and e88512[i]>=0 then ot[i]=-3; /* since there is no weeks per year. */
if e88501[i]>0 and e88502[i]=8  and e88512[i]>=0 then ot[i]=e88512[i]/(2.15*e88501[i]);
if e88501[i]>0 and e88502[i]=10 and hrwg[i]>=0 then ot[i]=hrwg[i]*1.5;
if e88501[i]>0 and e88502[i]=11 and hrwg[i]>=0 then ot[i]=hrwg[i]*2;
if e88502[i] in (9,14) then ot[i]=0;
if -4<e88501[i]<0 then ot[i]=e88501[i];
if e88512[i]>=0 and e88501[i]=0 then ot[i]=-3;
if e88502[i] in (12,13) and e88512[i]>=0 then do;
    /* condition based on the time unit to finish per hrwg. */
      if e88512aaa[i]=1 and e88512aab[i]>0 then ot[i]=(e88512[i]/e88512aab[i])*60;
      if e88512aaa[i]=2 and e88512aab[i]>0 then ot[i]=e88512[i]/e88512aab[i];
      if e88512aaa[i]=3 and e88512aab[i]>0 and e88512b[i]>=0 and e88501[i]>0 then
        ot[i]=(e88512[i]/e88512aab[i])*e88512b[i]/e88501[i];
      if e88512aaa[i]=4 and e88512aab[i]>0 and e88501[i]>0 then ot[i]=(e88512[i]/e88512aab[i])/e88501[i];
      if e88512aaa[i]=5 and e88512aab[i]>0 and e88501[i]>0 then ot[i]=(e88512[i]/e88512aab[i])/(e88501[i]*4.3);
    /* missing value */
      if -4<e88512aaa[i]<0 or -4<e88512aab[i]<=0 then ot[i]=-3;
      if e88512aaa[i] in (3,4,5) and -4<e88501[i]<=0 then ot[i]=-3;
      if e88512aaa[i]=3 and e88512b[i]<0 then ot[i]=-3;
   end;

/*missing value*/;
if -4<e88512[i]<0 then ot[i]=e88512[i];
if e88502[i]=2 and e88512b[i]=0 then ot[i]=-3;
if e88502[i]=2 and -4<e88512b[i]<0 then ot[i]=e88512b[i];

end;

do i=1 to 9;

if e98404a[i]=0 then multot=multot+1;
if e98403[i]>0 and e98404a[i]=1  and e98414[i]>=0 then ot[i]=e98414[i];
```

```
if e98403[i]>0 and e98404a[i]=2  and e98414b[i]>0 and e98414[i]>=0 then ot[i]=e98414[i]*e98414b[i]/e98403[i];
if e98403[i]>0 and e98404a[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2)  and e98414[i]>=0 then
            ot[i]=e98414[i]/e98403[i];
if e98403[i]>0 and e98404a[i]=4  and e98414[i]>=0 then ot[i]=e98414[i]/(2*e98403[i]);
if e98403[i]>0 and e98404a[i]=5  and e98414[i]>=0 then ot[i]=e98414[i]/(4.3*e98403[i]);
if e98403[i]>0 and e98404a[i]=6  and e98414[i]>=0 then ot[i]=-3; /* since there is no weeks per year. */
if e98403[i]>0 and e98404a[i]=8  and e98414[i]>=0 then ot[i]=e98414[i]/(2.15*e98403[i]);
if e98403[i]>0 and e98404a[i]=10 and hrwg[i]>=0 then ot[i]=hrwg[i]*1.5;
if e98403[i]>0 and e98404a[i]=11 and hrwg[i]>=0 then ot[i]=hrwg[i]*2;
if e98404a[i] in (9,14) then ot[i]=0;
if -4<e98403[i]<0 then ot[i]=e98403[i];
if e98414[i]>=0 and e98403[i]=0 then ot[i]=-3;
if e98404a[i] in (12,13) and e98414[i]>=0 then do;
    /* condition based on the time unit to finish per hrwg. */
      if e98414aaa[i]=1 and e98414aab[i]>0 then ot[i]=(e98414[i]/e98414aab[i])*60;
      if e98414aaa[i]=2 and e98414aab[i]>0 then ot[i]=e98414[i]/e98414aab[i];
      if e98414aaa[i]=3 and e98414aab[i]>0 and e98414b[i]>=0 and e98403[i]>0 then
        ot[i]=(e98414[i]/e98414aab[i])*e98414b[i]/e98403[i];
      if e98414aaa[i]=4 and e98414aab[i]>0 and e98403[i]>0 then ot[i]=(e98414[i]/e98414aab[i])/e98403[i];
      if e98414aaa[i]=5 and e98414aab[i]>0 and e98403[i]>0 then ot[i]=(e98414[i]/e98414aab[i])/(e98403[i]*4.3);
     /* missing value */
      if -4<e98414aaa[i]<0 or -4<e98414aab[i]<=0 then ot[i]=-3;
      if e98414aaa[i] in (3,4,5) and -4<e98403[i]<=0 then ot[i]=-3;
      if e98414aaa[i]=3 and e98414b[i]<0 then ot[i]=-3;
   end;

/*missing value*/;
if -4<e98414[i]<0 then ot[i]=e98414[i];
if e98404a[i]=2 and e98414b[i]=0 then ot[i]=-3;
if e98404a[i]=2 and -4<e98414b[i]<0 then ot[i]=e98414b[i];

end;

/**** report the corrected overtime payment if the correction is made *******/
do i=1 to 9;

if e226044[i]=1 and e226043[i]=0 then do; /*rate incorrect but hours correct*/
  if e19200[i]=1 then do;  /* report hourly payrate*/
    if e22612[i]=0 then multot=multot+1;
    if e22612[i]=1 and e22627[i]>=0 then ot[i]=e22627[i];
    if e22612[i]=2 and e22627[i]>=0 and e22627e[i]>0 and e24501[i]>0 then ot[i]=e22627[i]*e22627e[i]/e24501[i];
    if e22612[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28, 999,-1,-2) and e22627[i]>=0 and e24501[i]>0
            then ot[i]=e22627[i]/e24501[i];
    if e22612[i]=4 and e22627[i]>=0 and e24501[i]>0 then ot[i]=e22627[i]/(2*e24501[i]);
    if e22612[i]=5 and e22627[i]>=0 and e24501[i]>0 then ot[i]=e22627[i]/(4.3*e24501[i]);
    if e22612[i]=6 and e22627[i]>=0 and e24501[i]>0 then ot[i]=-3; /* since there is no weeks per year. */
    if e22612[i]=8 and e22627[i]>=0 and e24501[i]>0 then ot[i]=e22627[i]/(2.15*e24501[i]);
    if e22612[i] in (9,14) then ot[i]=0;
    if e22612[i] in (12,13) and e22627[i]>=0 then do;
      /* condition based on the time unit to finish per ot. */
      if e22627c[i]=1 and e22627cb[i]>0 then ot[i]=(e22627[i]/e22627cb[i])*60;
      if e22627c[i]=2 and e22627cb[i]>0 then ot[i]=e22627[i]/e22627cb[i];
      if e22627c[i]=3 and e22627cb[i]>0 and e22627e[i]>=0 and e24501[i]>0 then
        ot[i]=(e22627[i]/e22627cb[i])*e22627e[i]/e24501[i];
      if e22627c[i]=4 and e22627cb[i]>0 and e24501[i]>0 then ot[i]=(e22627[i]/e22627cb[i])/e24501[i];
      if e22627c[i]=5 and e22627cb[i]>0 and e24501[i]>0 then ot[i]=(e22627[i]/e22627cb[i])/(e24501[i]*4.3);
     /* missing value */
```

```
      if -4<e22627c[i]<0 or -4<e22627cb[i]<=0 then ot[i]=-3;
      if e22627c[i] in (3,4,5) and -4<e24501[i]<=0 then ot[i]=-3;
      if e22627c[i]=3 and e22627e[i]<0 then ot[i]=-3;
   end;
   if e22612[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e22627[i]<0 then
          ot[i]=e22627[i];
   if e22612[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e24501[i]=0 then ot[i]=-3;
   if e22612[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e24501[i]<0 then
          ot[i]=e24501[i];
    if e22612[i]=2 and e22627e[i] le 0 then ot[i]=-3;
    if e22612[i]=2 and -4<e22627e[i]<0 then ot[i]=e22627e[i];
  end;

  if e19200[i] ne 1 then do; /*report non-hourly payrate*/
    if e22612[i]=0 then multot=multot+1;
    if e22612[i]=1 and e22627[i]>=0 then ot[i]=e22627[i];
    if e22612[i]=2 and e22627[i]>=0 and e22627e[i]>0 and e34403[i]>0 then ot[i]=e22627[i]*e22627e[i]/e34403[i];
    if e22612[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28, 999,-1,-2) and e22627[i]>=0 and e34403[i]>0
          then ot[i]=e22627[i]/e34403[i];
    if e22612[i]=4 and e22627[i]>=0 and e34403[i]>0 then ot[i]=e22627[i]/(2*e34403[i]);
    if e22612[i]=5 and e22627[i]>=0 and e34403[i]>0 then ot[i]=e22627[i]/(4.3*e34403[i]);
    if e22612[i]=6 and e22627[i]>=0 and e34403[i]>0 then ot[i]=-3; /* since there is no weeks per year.*/
    if e22612[i]=8 and e22627[i]>=0 and e34403[i]>0 then ot[i]=e22627[i]/(2.15*e34403[i]);
    if e22612[i] in (9,14) then ot[i]=0;
    if e22612[i] in (12,13) and e22627[i]>=0 then do;
      /* condition based on the time unit to finish per ot. */
      if e22627c[i]=1 and e22627cb[i]>0 then ot[i]=(e22627[i]/e22627cb[i])*60;
      if e22627c[i]=2 and e22627cb[i]>0 then ot[i]=e22627[i]/e22627cb[i];
      if e22627c[i]=3 and e22627cb[i]>0 and e22627e[i]>=0 and e34403[i]>0 then
        ot[i]=(e22627[i]/e22627cb[i])*e22627e[i]/e34403[i];
      if e22627c[i]=4 and e22627cb[i]>0 and e34403[i]>0 then ot[i]=(e22627[i]/e22627cb[i])/e34403[i];
      if e22627c[i]=5 and e22627cb[i]>0 and e34403[i]>0 then ot[i]=(e22627[i]/e22627cb[i])/(e34403[i]*4.3);
     /* missing value */
      if -4<e22627c[i]<0 or -4<e22627cb[i]<=0 then ot[i]=-3;
      if e22627c[i] in (3,4,5) and -4<e34403[i]<=0 then ot[i]=-3;
      if e22627c[i]=3 and e22627e[i]<0 then ot[i]=-3;
   end;
   if e22612[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e22627[i]<0 then
          ot[i]=e22627[i];
   if e22612[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34403[i]=0 then ot[i]=-3;
   if e22612[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34403[i]<0 then
          ot[i]=e34403[i];
    if e22612[i]=2 and e22627e[i] le 0 then ot[i]=-3;
    if e22612[i]=2 and -4<e22627e[i]<0 then ot[i]=e22627e[i];
  end;
end;

if e226044[i]=0 and e226043[i]=1 then do; /*rate correct but hours incorrect*/
    if e24502[i] ne 1 and ot[i]>=0 and e34403[i]>0 and e22611[i]>0 then ot[i]=ot[i]*e34403[i]/e22611[i];
    if e34404[i] ne 1 and ot[i]>=0 and e34403[i]>0 and e22611[i]>0 then ot[i]=ot[i]*e34403[i]/e22611[i];
    if e24502[i] ne 1 and e34404[i] ne 1 and e22611[i]=0 then ot[i]=-3;
    if e24502[i] ne 1 and e34404[i] ne 1 and -4<e22611[i]<0 then ot[i]=e22611[i];
end;

if e226044[i]=1 and e226043[i]=1 then do; /* neither rate nor the hours is correct*/
  if e22612[i]=0 then multot=multot+1;
  if e22612[i]=1 and e22627[i]>=0 then ot[i]=e22627[i];
```

```
   if e22612[i]=2 and e22627[i]>=0 and e22627e[i]>0 and e22611[i]>0 then ot[i]=e22627[i]*e22627e[i]/e22611[i];
   if e22612[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28, 999,-1,-2) and e22627[i]>=0 and e22611[i]>0 then
            ot[i]=e22627[i]/e22611[i];
   if e22612[i]=4 and e22627[i]>=0 and e22611[i]>0 then ot[i]=e22627[i]/(2*e22611[i]);
   if e22612[i]=5 and e22627[i]>=0 and e22611[i]>0 then ot[i]=e22627[i]/(4.3*e22611[i]);
   if e22612[i]=6 and e22627[i]>=0 and e22611[i]>0 then ot[i]=-3; /* since there is no weeks per year. */
   if e22612[i]=8 and e22627[i]>=0 and e22611[i]>0 then ot[i]=e22627[i]/(2.15*e22611[i]);
   if e22612[i] in (9,14) then ot[i]=0;
   if e22612[i] in (12,13) and e22627[i]>=0 then do;
      /* condition based on the time unit to finish per ot. */
       if e22627c[i]=1 and e22627cb[i]>0 then ot[i]=(e22627[i]/e22627cb[i])*60;
       if e22627c[i]=2 and e22627cb[i]>0 then ot[i]=e22627[i]/e22627cb[i];
       if e22627c[i]=3 and e22627cb[i]>0 and e22627e[i]>=0 and e22611[i]>0 then
         ot[i]=(e22627[i]/e22627cb[i])*e22627e[i]/e22611[i];
       if e22627c[i]=4 and e22627cb[i]>0 and e22611[i]>0 then ot[i]=(e22627[i]/e22627cb[i])/e22611[i];
       if e22627c[i]=5 and e22627cb[i]>0 and e22611[i]>0 then ot[i]=(e22627[i]/e22627cb[i])/(e22611[i]*4.3);
     /* missing value */
       if -4<e22627c[i]<0 or -4<e22627cb[i]<=0 then ot[i]=-3;
       if e22627c[i] in (3,4,5) and -4<e22611[i]<=0 then ot[i]=-3;
       if e22627c[i]=3 and e22627e[i]<0 then ot[i]=-3;
    end;
    if e22612[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e22627[i]<0 then
            ot[i]=e22627[i];
    if e22612[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e22611[i]=0 then ot[i]=-3;
    if e22612[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e22611[i]<0 then
            ot[i]=e22611[i];
     if e22612[i]=2 and e22627e[i] le 0 then ot[i]=-3;
     if e22612[i]=2 and -4<e22627e[i]<0 then ot[i]=e22627e[i];

end;
end;

/**** correction made in the later part ****/

do i=1 to 9;

if e102254[i]=1 and e102253[i]=0 then do; /*rate incorrect but hours correct*/
   if e83100[i]=1 then do;  /* report hourly payrate*/
     if e100233[i]=0 then multot=multot+1;
     if e100233[i]=1 and e100249[i]>=0 then ot[i]=e100249[i];
     if e100233[i]=2 and e100249[i]>=0 and e100249e[i]>0 and e88501[i]>0 then
            ot[i]=e100249[i]*e100249e[i]/e88501[i];
     if e100233[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28, 999,-1,-2) and e100249[i]>=0 and e88501[i]>0
            then ot[i]=e100249[i]/e88501[i];
     if e100233[i]=4 and e100249[i]>=0 and e88501[i]>0 then ot[i]=e100249[i]/(2*e88501[i]);
     if e100233[i]=5 and e100249[i]>=0 and e88501[i]>0 then ot[i]=e100249[i]/(4.3*e88501[i]);
     if e100233[i]=6 and e100249[i]>=0 and e88501[i]>0 then ot[i]=-3; /* since there is no weeks per year. */
     if e100233[i]=8 and e100249[i]>=0 and e88501[i]>0 then ot[i]=e100249[i]/(2.15*e88501[i]);
     if e100233[i] in (9,14) then ot[i]=0;
     if e100233[i] in (12,13) and e100249[i]>=0 then do;
       /* condition based on the time unit to finish per ot. */
        if e100249c[i]=1 and e100249cb[i]>0 then ot[i]=(e100249[i]/e100249cb[i])*60;
        if e100249c[i]=2 and e100249cb[i]>0 then ot[i]=e100249[i]/e100249cb[i];
        if e100249c[i]=3 and e100249cb[i]>0 and e100249e[i]>=0 and e88501[i]>0 then
          ot[i]=(e100249[i]/e100249cb[i])*e100249e[i]/e88501[i];
        if e100249c[i]=4 and e100249cb[i]>0 and e88501[i]>0 then ot[i]=(e100249[i]/e100249cb[i])/e88501[i];
        if e100249c[i]=5 and e100249cb[i]>0 and e88501[i]>0 then ot[i]=(e100249[i]/e100249cb[i])/(e88501[i]*4.3);
```

```
    /* missing value */
      if -4<e100249c[i]<0 or -4<e100249cb[i]<=0 then ot[i]=-3;
      if e100249c[i] in (3,4,5) and -4<e88501[i]<=0 then ot[i]=-3;
      if e100249c[i]=3 and e100249e[i]<0 then ot[i]=-3;
  end;
  if e100233[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e100249[i]<0 then
          ot[i]=e100249[i];
  if e100233[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e88501[i]=0 then ot[i]=-3;
  if e100233[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e88501[i]<0 then
          ot[i]=e88501[i];
   if e100233[i]=2 and e100249e[i] le 0 then ot[i]=-3;
   if e100233[i]=2 and -4<e100249e[i]<0 then ot[i]=e100249e[i];

 end;

 if e83100[i] ne 1 then do; /*report non-hourly payrate*/
   if e100233[i]=0 then multot=multot+1;
   if e100233[i]=1 and e100249[i]>=0 then ot[i]=e100249[i];
   if e100233[i]=2 and e100249[i]>=0 and e100249e[i]>0 and e98403[i]>0 then
          ot[i]=e100249[i]*e100249e[i]/e98403[i];
   if e100233[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28, 999,-1,-2) and e100249[i]>=0 and e98403[i]>0
          then ot[i]=e100249[i]/e98403[i];
   if e100233[i]=4 and e100249[i]>=0 and e98403[i]>0 then ot[i]=e100249[i]/(2*e98403[i]);
   if e100233[i]=5 and e100249[i]>=0 and e98403[i]>0 then ot[i]=e100249[i]/(4.3*e98403[i]);
   if e100233[i]=6 and e100249[i]>=0 and e98403[i]>0 then ot[i]=-3; /* since there is no weeks per year.*/
   if e100233[i]=8 and e100249[i]>=0 and e98403[i]>0 then ot[i]=e100249[i]/(2.15*e98403[i]);
   if e100233[i] in (9,14) then ot[i]=0;
   if e100233[i] in (12,13) and e100249[i]>=0 then do;
     /* condition based on the time unit to finish per ot. */
      if e100249c[i]=1 and e100249cb[i]>0 then ot[i]=(e100249[i]/e100249cb[i])*60;
      if e100249c[i]=2 and e100249cb[i]>0 then ot[i]=e100249[i]/e100249cb[i];
      if e100249c[i]=3 and e100249cb[i]>0 and e100249e[i]>=0 and e98403[i]>0 then
        ot[i]=(e100249[i]/e100249cb[i])*e100249e[i]/e98403[i];
      if e100249c[i]=4 and e100249cb[i]>0 and e98403[i]>0 then ot[i]=(e100249[i]/e100249cb[i])/e98403[i];
      if e100249c[i]=5 and e100249cb[i]>0 and e98403[i]>0 then ot[i]=(e100249[i]/e100249cb[i])/(e98403[i]*4.3);
    /* missing value */
      if -4<e100249c[i]<0 or -4<e100249cb[i]<=0 then ot[i]=-3;
      if e100249c[i] in (3,4,5) and -4<e98403[i]<=0 then ot[i]=-3;
      if e100249c[i]=3 and e100249e[i]<0 then ot[i]=-3;
  end;
  if e100233[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e100249[i]<0 then
          ot[i]=e100249[i];
  if e100233[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98403[i]=0 then ot[i]=-3;
  if e100233[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98403[i]<0 then
          ot[i]=e98403[i];
   if e100233[i]=2 and e100249e[i] le 0 then ot[i]=-3;
   if e100233[i]=2 and -4<e100249e[i]<0 then ot[i]=e100249e[i];
 end;
end;

if e102254[i]=0 and e102253[i]=1 then do; /*rate correct but hours incorrect*/
   if e88502[i] ne 1 and ot[i]>=0 and e88501[i]>0 and e100232[i]>0 then ot[i]=ot[i]*e88501[i]/e100232[i];
   if e98404a[i] ne 1 and ot[i]>=0 and e98403[i]>0 and e100232[i]>0 then ot[i]=ot[i]*e98403[i]/e100232[i];
   if e88502[i] ne 1 and e98404a[i] ne 1 and e100232[i]=0 then ot[i]=-3;
   if e88502[i] ne 1 and e98404a[i] ne 1 and -4<e100232[i]<0 then ot[i]=e100232[i];
end;
```

```
if e102254[i]=1 and e102253[i]=1 then do; /* neither rate nor the hours is correct*/
   if e100233[i]=0 then multot=multot+1;
   if e100233[i]=1 and e100249[i]>=0 then ot[i]=e100249[i];
   if e100233[i]=2 and e100249[i]>=0 and e100249e[i]>0 and e100232[i]>0 then
            ot[i]=e100249[i]*e100249e[i]/e100232[i];
   if e100233[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28, 999,-1,-2) and e100249[i]>=0 and e100232[i]>0
            then ot[i]=e100249[i]/e100232[i];
   if e100233[i]=4 and e100249[i]>=0 and e100232[i]>0 then ot[i]=e100249[i]/(2*e100232[i]);
   if e100233[i]=5 and e100249[i]>=0 and e100232[i]>0 then ot[i]=e100249[i]/(4.3*e100232[i]);
   if e100233[i]=6 and e100249[i]>=0 and e100232[i]>0 then ot[i]=-3; /* since there is no weeks per year. */
   if e100233[i]=8 and e100249[i]>=0 and e100232[i]>0 then ot[i]=e100249[i]/(2.15*e100232[i]);
   if e100233[i] in (9,14) then ot[i]=0;
   if e100233[i] in (12,13) and e100249[i]>=0 then do;
      /* condition based on the time unit to finish per ot. */
       if e100249c[i]=1 and e100249cb[i]>0 then ot[i]=(e100249[i]/e100249cb[i])*60;
       if e100249c[i]=2 and e100249cb[i]>0 then ot[i]=e100249[i]/e100249cb[i];
       if e100249c[i]=3 and e100249cb[i]>0 and e100249e[i]>=0 and e100232[i]>0 then
         ot[i]=(e100249[i]/e100249cb[i])*e100249e[i]/e100232[i];
       if e100249c[i]=4 and e100249cb[i]>0 and e100232[i]>0 then ot[i]=(e100249[i]/e100249cb[i])/e100232[i];
       if e100249c[i]=5 and e100249cb[i]>0 and e100232[i]>0 then ot[i]=(e100249[i]/e100249cb[i])/(e100232[i]*4.3);
      /* missing value */
       if -4<e100249c[i]<0 or -4<e100249cb[i]<=0 then ot[i]=-3;
       if e100249c[i] in (3,4,5) and -4<e100232[i]<=0 then ot[i]=-3;
       if e100249c[i]=3 and e100249e[i]<0 then ot[i]=-3;
    end;
    if e100233[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e100249[i]<0 then
            ot[i]=e100249[i];
    if e100233[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e100232[i]=0 then ot[i]=-3;
    if e100233[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e100232[i]<0 then
            ot[i]=e100232[i];
     if e100233[i]=2 and e100249e[i] le 0 then ot[i]=-3;
     if e100233[i]=2 and -4<e100249e[i]<0 then ot[i]=e100249e[i];

  end;
end;

/* amount paid on tips, commissions, bonuses, incentive pay or other*/
array othpay1 othpay11 othpay21 othpay31 othpay41 othpay51 othpay61 othpay71 othpay81 othpay91;
array othpay2 othpay12 othpay22 othpay32 othpay42 othpay52 othpay62 othpay72 othpay82 othpay92;
array othpay3 othpay13 othpay23 othpay33 othpay43 othpay53 othpay63 othpay73 othpay83 othpay93;
array othpay4 othpay14 othpay24 othpay34 othpay44 othpay54 othpay64 othpay74 othpay84 othpay94;
array othpay5 othpay15 othpay25 othpay35 othpay45 othpay55 othpay65 othpay75 othpay85 othpay95;

array othpf1 othpf11 othpf21 othpf31 othpf41 othpf51 othpf61 othpf71 othpf81 othpf91;
array othpf2 othpf12 othpf22 othpf32 othpf42 othpf52 othpf62 othpf72 othpf82 othpf92;
array othpf3 othpf13 othpf23 othpf33 othpf43 othpf53 othpf63 othpf73 othpf83 othpf93;
array othpf4 othpf14 othpf24 othpf34 othpf44 othpf54 othpf64 othpf74 othpf84 othpf94;
array othpf5 othpf15 othpf25 othpf35 othpf45 othpf55 othpf65 othpf75 othpf85 othpf95;

do i=1 to 9;
   othpay1[i]=-4;   othpay2[i]=-4;   othpay3[i]=-4;   othpay4[i]=-4;   othpay5[i]=-4;
   othpf1[i]=0;     othpf2[i]=0;     othpf3[i]=0;     othpf4[i]=0;     othpf5[i]=0;
end;

/* do it for non-overtime payment*/

/********* for tips ***********/
```

```
/*report hourly wage at the beginning*/
do i=1 to 9;
if e23901[i]>0 and e216001[i]=1 and e225001[i]>=0 then othpay1[i]=e225001[i];
if e23901[i]>0 and e216001[i]=2 and e225501[i] and e225001[i]>=0 then
        othpay1[i]=(e225001[i]*e225501[i])/e23901[i];
if e23901[i]>0 and e216001[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e225001[i]>=0
        then othpay1[i]=e225001[i]/e23901[i];
if e23901[i]>0 and e216001[i]=4 and e225001[i]>=0 then othpay1[i]=e225001[i]/(2*e23901[i]);
if e23901[i]>0 and e216001[i]=5 and e225001[i]>=0 then othpay1[i]=e225001[i]/(4.3*e23901[i]);
if e23901[i]>0 and e216001[i]=6 and e225001[i]>=0 and e35600[i]>0 then othpay1[i]=e225001[i]/(e35600[i]
        *e23901[i]);
if e23901[i]>0 and e216001[i]=8 and e225001[i]>=0 then othpay1[i]=e225001[i]/(2.15*e23901[i]);
if e216001[i] in (9,14) then othpay1[i]=0;
if e216001[i] in (12,13) and e225001[i]>=0 then do;
   /* condition based on the time unit to finish per othpay1. */
    if e225301[i]=1 and e225351[i]>0 then othpay1[i]=(e225001[i]/e225351[i])*60;
    if e225301[i]=2 and e225351[i]>0 then othpay1[i]=e225001[i]/e225351[i];
    if e225301[i]=3 and e225351[i]>0 and e225501[i]>=0 and e23901[i]>0 then
      othpay1[i]=(e225001[i]/e225351[i])*e225501[i]/e23901[i];
    if e225301[i]=4 and e225351[i]>0 and e23901[i]>0 then othpay1[i]=(e225001[i]/e225351[i])/e23901[i];
    if e225301[i]=5 and e225351[i]>0 and e23901[i]>0 then othpay1[i]=(e225001[i]/e225351[i])/(e23901[i]*4.3);
  /* missing value */
    if -4<e225301[i]<0 or -4<e225351[i]<=0 then othpay1[i]=-3;
    if e225301[i] in (3,4,5) and -4<e23901[i]<=0 then othpay1[i]=-3;
    if e225301[i]=3 and e225501[i]<0 then othpay1[i]=-3;
end;
 if e216001[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e225001[i]<0 then
        othpay1[i]=e225001[i];
 if e216001[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay1[i]=-3;
 if e216001[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay1[i]=e23901[i];
 if e216001[i]=2 and e225501[i] le 0 then othpay1[i]=-3;
 if e216001[i]=2 and -4<e225501[i]<0 then othpay1[i]=e225501[i];

end;

/*non-hourly wage at the beginning, with overtime*/
do i=1 to 9;
if e34428[i]>0 and e216001[i]=1 and e225001[i]>=0 then othpay1[i]=e225001[i];
if e34428[i]>0 and e216001[i]=2 and e34430[i]>0 and e225001[i]>=0 then
        othpay1[i]=(e225001[i]*e34430[i])/e34428[i];
if e34428[i]>0 and e216001[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e225001[i]>=0
        then othpay1[i]=e225001[i]/e34428[i];
if e34428[i]>0 and e216001[i]=4 and e225001[i]>=0 then othpay1[i]=e225001[i]/(2*e34428[i]);
if e34428[i]>0 and e216001[i]=5 and e225001[i]>=0 then othpay1[i]=e225001[i]/(4.3*e34428[i]);
if e34428[i]>0 and e216001[i]=6 and e225001[i]>=0 and e35600[i]>0 then othpay1[i]=e225001[i]/(e35600[i]
        *e34428[i]);
if e34428[i]>0 and e216001[i]=8 and e225001[i]>=0 then othpay1[i]=e225001[i]/(2.15*e34428[i]);
if e216001[i] in (12,13) and e225001[i]>=0 then do;
   /* condition based on the time unit to finish per othpay1. */
    if e225301[i]=3 and e225351[i]>0 and e225501[i]>=0 and e34428[i]>0 then
      othpay1[i]=(e225001[i]/e225351[i])*e225501[i]/e34428[i];
    if e225301[i]=4 and e225351[i]>0 and e34428[i]>0 then othpay1[i]=(e225001[i]/e225351[i])/e34428[i];
    if e225301[i]=5 and e225351[i]>0 and e34428[i]>0 then othpay1[i]=(e225001[i]/e225351[i])/(e34428[i]*4.3);
  /* missing value */
    if e225301[i] in (3,4,5) and -4<e34428[i]<=0 then othpay1[i]=-3;
```

```
  end;
   if e216001[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay1[i]=-3;
   if e216001[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
            othpay1[i]=e34428[i];
end;

  /* non-hourly wage at the beginning, without overtime*/
  do i=1 to 9;
  if e34402[i]>0 and e216001[i]=1 and e225001[i]>=0 then othpay1[i]=e225001[i];
  if e34402[i]>0 and e216001[i]=2 and e225501[i]>0 and e225001[i]>=0 then
            othpay1[i]=(e225001[i]*e225501[i])/e34402[i];
  if e34402[i]>0 and e216001[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e225001[i]>=0
            then othpay1[i]=e225001[i]/e34402[i];
  if e34402[i]>0 and e216001[i]=4 and e225001[i]>=0 then othpay1[i]=e225001[i]/(2*e34402[i]);
  if e34402[i]>0 and e216001[i]=5 and e225001[i]>=0 then othpay1[i]=e225001[i]/(4.3*e34402[i]);
  if e34402[i]>0 and e216001[i]=6 and e225001[i]>=0 and e35600[i]>0 then
            othpay1[i]=e225001[i]/(e35600[i]*e34402[i]);
  if e34402[i]>0 and e216001[i]=8 and e225001[i]>=0 then othpay1[i]=e225001[i]/(2.15*e34402[i]);
  if e216001[i] in (7,9,12,13,14,15,16,17,18,19,21,22,23,24,25,26,28) then othpf1[i]=othpf1[i]+1;
  if e216001[i] in (12,13) and e225001[i]>=0 then do;
     /* condition based on the time unit to finish per othpay1. */
      if e225301[i]=3 and e225351[i]>0 and e225501[i]>=0 and e34402[i]>0 then
        othpay1[i]=(e225001[i]/e225351[i])*e225501[i]/e34402[i];
      if e225301[i]=4 and e225351[i]>0 and e34402[i]>0 then othpay1[i]=(e225001[i]/e225351[i])/e34402[i];
      if e225301[i]=5 and e225351[i]>0 and e34402[i]>0 then othpay1[i]=(e225001[i]/e225351[i])/(e34402[i]*4.3);
    /* missing value */
      if e225301[i] in (3,4,5) and -4<e34402[i]<=0 then othpay1[i]=-3;
  end;
   if e216001[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay1[i]=-3;
   if e216001[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
            othpay1[i]=e34402[i];

  if e216001[i]=6 and e35600[i] le 0 then othpay1[i]=-3;
  if e216001[i]=6 and -4<e35600[i]<0 then othpay1[i]=e35600[i];
  end;

  /* if tips is corrected in the later part */
  do i=1 to 9;
  if e226045[i]=1 then do;
  /*report hourly wage at the beginning*/
  if e22613[i]=1 and e22628[i]>=0 then othpay1[i]=e22628[i];
  if e22613[i]=2 and e22628[i]>=0 and e34430[i]>0 and e23901[i]>0 then othpay1[i]=e22628[i]*e34430[i]/e23901[i];
  if e22613[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28, 999,-1,-2)  and e22628[i]>=0 and e23901[i]>0 then
            othpay1[i]=e22628[i]/e23901[i];
  if e22613[i]=4 and e22628[i]>=0 and e23901[i]>0 then othpay1[i]=e22628[i]/(2*e23901[i]);
  if e22613[i]=5 and e22628[i]>=0 and e23901[i]>0 then othpay1[i]=e22628[i]/(4.3*e23901[i]);
  if e22613[i]=6 and e22628[i]>=0 and e23901[i]>0 and e35600[i]>0 then othpay1[i]=e22628[i]/(e35600[i]*e23901[i]);
  if e22613[i]=8 and e22628[i]>=0 and e23901[i]>0 then othpay1[i]=e22628[i]/(2.15*e23901[i]);
  if e22613[i] in (9,14) then othpay1[i]=0;
  if e22613[i] in (12,13) and e22628[i]>=0 then do;
     /* condition based on the time unit to finish per othpay1. */
      if e22628c[i]=1 and e22628cb[i]>0 then othpay1[i]=(e22628[i]/e22628cb[i])*60;
      if e22628c[i]=2 and e22628cb[i]>0 then othpay1[i]=e22628[i]/e22628cb[i];
      if e22628c[i]=3 and e22628cb[i]>0 and e22628e[i]>=0 and e23901[i]>0 then
        othpay1[i]=(e22628[i]/e22628cb[i])*e22628e[i]/e23901[i];
      if e22628c[i]=4 and e22628cb[i]>0 and e23901[i]>0 then othpay1[i]=(e22628[i]/e22628cb[i])/e23901[i];
      if e22628c[i]=5 and e22628cb[i]>0 and e23901[i]>0 then othpay1[i]=(e22628[i]/e22628cb[i])/(e23901[i]*4.3);
```

```
  /* missing value */
    if -4<e22628c[i]<0 or -4<e22628cb[i]<=0 then othpay1[i]=-3;
    if e22628c[i] in (3,4,5) and -4<e23901[i]<=0 then othpay1[i]=-3;
    if e22628c[i]=3 and e22628e[i]<0 then othpay1[i]=-3;
end;
if e22613[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e22628[i]<0 then
        othpay1[i]=e22628[i];
if e22613[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay1[i]=-3;
if e22613[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay1[i]=e23901[i];
if e22613[i]=2 and e22628e[i] le 0 then othpay1[i]=-3;
if e22613[i]=2 and -4<e22628e[i]<0 then othpay1[i]=e22628e[i];

/* non-hourly wage at the beginning, with overtime*/
if e22613[i]=2 and e22628[i]>=0 and e22628e[i]>0 and e34428[i]>0 then othpay1[i]=e22628[i]*e22628e[i]/e34428[i];
if e22613[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28, 999,-1,-2) and e22628[i]>=0 and e34428[i]>0 then
        othpay1[i]=e22628[i]/e34428[i];
if e22613[i]=4 and e22628[i]>=0 and e34428[i]>0 then othpay1[i]=e22628[i]/(2*e34428[i]);
if e22613[i]=5 and e22628[i]>=0 and e34428[i]>0 then othpay1[i]=e22628[i]/(4.3*e34428[i]);
if e22613[i]=6 and e22628[i]>=0 and e34428[i]>0 and e35600[i]>0 then othpay1[i]=e22628[i]/(e35600[i]*e34428[i]);
if e22613[i]=8 and e22628[i]>=0 and e34428[i]>0 then othpay1[i]=e22628[i]/(2.15*e34428[i]);
if e22613[i] in (12,13) and e22628[i]>=0 then do;
   /* condition based on the time unit to finish per othpay1. */
    if e22628c[i]=3 and e22628cb[i]>0 and e22628e[i]>=0 and e34428[i]>0 then
      othpay1[i]=(e22628[i]/e22628cb[i])*e22628e[i]/e34428[i];
    if e22628c[i]=4 and e22628cb[i]>0 and e34428[i]>0 then othpay1[i]=(e22628[i]/e22628cb[i])/e34428[i];
    if e22628c[i]=5 and e22628cb[i]>0 and e34428[i]>0 then othpay1[i]=(e22628[i]/e22628cb[i])/(e34428[i]*4.3);
  /* missing value */
    if e22628c[i] in (3,4,5) and -4<e34428[i]<=0 then othpay1[i]=-3;
end;
if e22613[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay1[i]=-3;
if e22613[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay1[i]=e34428[i];

/* non-hourly wage at the beginning, without overtime*/
if e22613[i]=2 and e22628[i]>=0 and e22628e[i]>0 and e34402[i]>0 then othpay1[i]=e22628[i]*e22628e[i]/e34402[i];
if e22613[i] in (3, 7, 15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28, 999,-1,-2) and e22628[i]>=0 and e34402[i]>0 then
        othpay1[i]=e22628[i]/e34402[i];
if e22613[i]=4 and e22628[i]>=0 and e34402[i]>0 then othpay1[i]=e22628[i]/(2*e34402[i]);
if e22613[i]=5 and e22628[i]>=0 and e34402[i]>0 then othpay1[i]=e22628[i]/(4.3*e34402[i]);
if e22613[i]=6 and e22628[i]>=0 and e34402[i]>0 and e35600[i]>0 then othpay1[i]=e22628[i]/(e35600[i]*e34402[i]);
if e22613[i]=8 and e22628[i]>=0 and e34402[i]>0 then othpay1[i]=e22628[i]/(2.15*e34402[i]);
if e22613[i] in (12,13) and e22628[i]>=0 then do;
   /* condition based on the time unit to finish per othpay1. */
    if e22628c[i]=3 and e22628cb[i]>0 and e22628e[i]>=0 and e34402[i]>0 then
      othpay1[i]=(e22628[i]/e22628cb[i])*e22628e[i]/e34402[i];
    if e22628c[i]=4 and e22628cb[i]>0 and e34402[i]>0 then othpay1[i]=(e22628[i]/e22628cb[i])/e34402[i];
    if e22628c[i]=5 and e22628cb[i]>0 and e34402[i]>0 then othpay1[i]=(e22628[i]/e22628cb[i])/(e34402[i]*4.3);
  /* missing value */
    if e22628c[i] in (3,4,5) and -4<e34402[i]<=0 then othpay1[i]=-3;
end;
if e22613[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay1[i]=-3;
if e22613[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
        othpay1[i]=e34402[i];

if e22613[i]=6 and e35600[i] le 0 then othpay1[i]=-3;
if e22613[i]=6 and -4<e35600[i]<0 then othpay1[i]=e35600[i];
```

```
  end;
  end;

/********* for commissions **********/

   /*report hourly wage at the beginning*/
   do i=1 to 9;
   if e216002[i]=1 and e225002[i]>=0 then othpay2[i]=e225002[i];
   if e23901[i]>0 and e216002[i]=2 and e225502[i]>0 and e225002[i]>=0 then
            othpay2[i]=(e225002[i]*e225502[i])/e23901[i];
   if e23901[i]>0 and e216002[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e225002[i]>=0
            then othpay2[i]=e225002[i]/e23901[i];
   if e23901[i]>0 and e216002[i]=4 and e225002[i]>=0 then othpay2[i]=e225002[i]/(2*e23901[i]);
   if e23901[i]>0 and e216002[i]=5 and e225002[i]>=0 then othpay2[i]=e225002[i]/(4.3*e23901[i]);
   if e23901[i]>0 and e216002[i]=6 and e225002[i]>=0 and e35600[i]>0 then
            othpay2[i]=e225002[i]/(e35600[i]*e23901[i]);
   if e23901[i]>0 and e216002[i]=8 and e225002[i]>=0 then othpay2[i]=e225002[i]/(2.15*e23901[i]);
   if e216002[i] in (9,14) then othpay2[i]=0;
   if e216002[i] in (12,13) and e225002[i]>=0 then do;
      /* condition based on the time unit to finish per othpay2. */
       if e225302[i]=1 and e225352[i]>0 then othpay2[i]=(e225002[i]/e225352[i])*60;
       if e225302[i]=2 and e225352[i]>0 then othpay2[i]=e225002[i]/e225352[i];
       if e225302[i]=3 and e225352[i]>0 and e225502[i]>=0 and e23901[i]>0 then
         othpay2[i]=(e225002[i]/e225352[i])*e225502[i]/e23901[i];
       if e225302[i]=4 and e225352[i]>0 and e23901[i]>0 then othpay2[i]=(e225002[i]/e225352[i])/e23901[i];
       if e225302[i]=5 and e225352[i]>0 and e23901[i]>0 then othpay2[i]=(e225002[i]/e225352[i])/(e23901[i]*4.3);
      /* missing value */
       if -4<e225302[i]<0 or -4<e225352[i]<=0 then othpay2[i]=-3;
       if e225302[i] in (3,4,5) and -4<e23901[i]<=0 then othpay2[i]=-3;
       if e225302[i]=3 and e225502[i]<0 then othpay2[i]=-3;
   end;
    if e216002[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e225002[i]<0 then
            othpay2[i]=e225002[i];
    if e216002[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay2[i]=-3;
    if e216002[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
            othpay2[i]=e23901[i];
    if e216002[i]=2 and e225502[i] le 0 then othpay2[i]=-3;
    if e216002[i]=2 and -4<e225502[i]<0 then othpay2[i]=e225502[i];

  end;

   /* non-hourly wage at the beginning,with overttme*/
   do i=1 to 9;
   if e34428[i]>0 and e216002[i]=2 and e225502[i]>0 and e225002[i]>=0 then
            othpay2[i]=(e225002[i]*e225502[i])/e34428[i];
   if e34428[i]>0 and e216002[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e225002[i]>=0
            then othpay2[i]=e225002[i]/e34428[i];
   if e34428[i]>0 and e216002[i]=4 and e225002[i]>=0 then othpay2[i]=e225002[i]/(2*e34428[i]);
   if e34428[i]>0 and e216002[i]=5 and e225002[i]>=0 then othpay2[i]=e225002[i]/(4.3*e34428[i]);
   if e34428[i]>0 and e216002[i]=6 and e225002[i]>=0 and e35600[i]>0 then
            othpay2[i]=e225002[i]/(e35600[i]*e34428[i]);
   if e34428[i]>0 and e216002[i]=8 and e225002[i]>=0 then othpay2[i]=e225002[i]/(2.15*e34428[i]);
   if e216002[i] in (12,13) and e225002[i]>=0 then do;
      /* condition based on the time unit to finish per othpay2. */
       if e225302[i]=3 and e225352[i]>0 and e225502[i]>=0 and e34428[i]>0 then
         othpay2[i]=(e225002[i]/e225352[i])*e225502[i]/e34428[i];
```

```
    if e225302[i]=4 and e225352[i]>0 and e34428[i]>0 then othpay2[i]=(e225002[i]/e225352[i])/e34428[i];
    if e225302[i]=5 and e225352[i]>0 and e34428[i]>0 then othpay2[i]=(e225002[i]/e225352[i])/(e34428[i]*4.3);
  /* missing value */
    if e225302[i] in (3,4,5) and -4<e34428[i]<=0 then othpay2[i]=-3;
end;
 if e216002[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay2[i]=-3;
 if e216002[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay2[i]=e34428[i];

end;

/* non-hourly wage at the beginning, without overtime*/
do i=1 to 9;
if e34402[i]>0 and e216002[i]=2 and e225502[i]>0 and e225002[i]>=0 then
        othpay2[i]=(e225002[i]*e225502[i])/e34402[i];
if e34402[i]>0 and e216002[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e225002[i]>=0
        then othpay2[i]=e225002[i]/e34402[i];
if e34402[i]>0 and e216002[i]=4 and e225002[i]>=0 then othpay2[i]=e225002[i]/(2*e34402[i]);
if e34402[i]>0 and e216002[i]=5 and e225002[i]>=0 then othpay2[i]=e225002[i]/(4.3*e34402[i]);
if e34402[i]>0 and e216002[i]=6 and e225002[i]>=0 and e35600[i]>0 then
        othpay2[i]=e225002[i]/(e35600[i]*e34402[i]);
if e34402[i]>0 and e216002[i]=8 and e225002[i]>=0 then othpay2[i]=e225002[i]/(2.15*e34402[i]);
if e216002[i] in (7,9,12,13,15,16,17,14,18,19,21,22,23,24,25,26,28) then othpf2[i]=othpf2[i]+1;
if e216002[i] in (12,13) and e225002[i]>=0 then do;
   /* condition based on the time unit to finish per othpay2. */
    if e225302[i]=3 and e225352[i]>0 and e225502[i]>=0 and e34402[i]>0 then
      othpay2[i]=(e225002[i]/e225352[i])*e225502[i]/e34402[i];
    if e225302[i]=4 and e225352[i]>0 and e34402[i]>0 then othpay2[i]=(e225002[i]/e225352[i])/e34402[i];
    if e225302[i]=5 and e225352[i]>0 and e34402[i]>0 then othpay2[i]=(e225002[i]/e225352[i])/(e34402[i]*4.3);
  /* missing value */
    if e225302[i] in (3,4,5) and -4<e34402[i]<=0 then othpay2[i]=-3;
end;
 if e216002[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay2[i]=-3;
 if e216002[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
        othpay2[i]=e34402[i];

if e216002[i]=6 and e35600[i] le 0 then othpay2[i]=-3;
if e216002[i]=6 and -4<e35600[i]<0 then othpay2[i]=e35600[i];
end;

/* if commissions is corrected in the later part */
do i=1 to 9;
if e226046[i]=1 then do;

/*report hourly wage at the beginning*/
if e22614[i]=1 and e22629[i]>=0 then othpay2[i]=e22629[i];
if e22614[i]=2 and e22629[i]>=0 and e22629e[i]>0 and e23901[i]>0 then othpay2[i]=e22629[i]*e22629e[i]/e23901[i];
if e22614[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e22629[i]>=0 and e23901[i]>0 then
        othpay2[i]=e22629[i]/e23901[i];
if e22614[i]=4 and e22629[i]>=0 and e23901[i]>0 then othpay2[i]=e22629[i]/(2*e23901[i]);
if e22614[i]=5 and e22629[i]>=0 and e23901[i]>0 then othpay2[i]=e22629[i]/(4.3*e23901[i]);
if e22614[i]=6 and e22629[i]>=0 and e23901[i]>0 and e35600[i]>0 then othpay2[i]=e22629[i]/(e35600[i]*e23901[i]);
if e22614[i]=8 and e22629[i]>=0 and e23901[i]>0 then othpay2[i]=e22629[i]/(2.15*e23901[i]);
if e22614[i] in (9,14) then othpay2[i]=0;
 if e22614[i] in (12,13) and e22629[i]>=0 then do;
   /* condition based on the time unit to finish per othpay2. */
    if e22629c[i]=1 and e22629cb[i]>0 then othpay2[i]=(e22629[i]/e22629cb[i])*60;
```

```
    if e22629c[i]=2 and e22629cb[i]>0 then othpay2[i]=e22629[i]/e22629cb[i];
    if e22629c[i]=3 and e22629cb[i]>0 and e22629e[i]>=0 and e23901[i]>0 then
      othpay2[i]=(e22629[i]/e22629cb[i])*e22629e[i]/e23901[i];
    if e22629c[i]=4 and e22629cb[i]>0 and e23901[i]>0 then othpay2[i]=(e22629[i]/e22629cb[i])/e23901[i];
    if e22629c[i]=5 and e22629cb[i]>0 and e23901[i]>0 then othpay2[i]=(e22629[i]/e22629cb[i])/(e23901[i]*4.3);
  /* missing value */
    if -4<e22629c[i]<0 or -4<e22629cb[i]<=0 then othpay2[i]=-3;
    if e22629c[i] in (3,4,5) and -4<e23901[i]<=0 then othpay2[i]=-3;
    if e22629c[i]=3 and e22629e[i]<0 then othpay2[i]=-3;
end;
if e22614[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e22629[i]<0 then
        othpay2[i]=e22629[i];
if e22614[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay2[i]=-3;
if e22614[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay2[i]=e23901[i];
if e22614[i]=2 and e22629e[i] le 0 then othpay2[i]=-3;
if e22614[i]=2 and -4<e22629e[i]<0 then othpay2[i]=e22629e[i];

/* non-hourly wage at the beginning, with overtime*/
if e22614[i]=2 and e22629[i]>=0 and e22629e[i]>0 and e34428[i]>0 then othpay2[i]=e22629[i]*e22629e[i]/e34428[i];
if e22614[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e22629[i]>=0 and e34428[i]>0 then
        othpay2[i]=e22629[i]/e34428[i];
if e22614[i]=4 and e22629[i]>=0 and e34428[i]>0 then othpay2[i]=e22629[i]/(2*e34428[i]);
if e22614[i]=5 and e22629[i]>=0 and e34428[i]>0 then othpay2[i]=e22629[i]/(4.3*e34428[i]);
if e22614[i]=6 and e22629[i]>=0 and e34428[i]>0 and e35600[i]>0 then othpay2[i]=e22629[i]/(e35600[i]*e34428[i]);
if e22614[i]=8 and e22629[i]>=0 and e34428[i]>0 then othpay2[i]=e22629[i]/(2.15*e34428[i]);
if e22614[i] in (12,13) and e22629[i]>=0 then do;
   /* condition based on the time unit to finish per othpay2. */
    if e22629c[i]=3 and e22629cb[i]>0 and e22629e[i]>=0 and e34428[i]>0 then
      othpay2[i]=(e22629[i]/e22629cb[i])*e22629e[i]/e34428[i];
    if e22629c[i]=4 and e22629cb[i]>0 and e34428[i]>0 then othpay2[i]=(e22629[i]/e22629cb[i])/e34428[i];
    if e22629c[i]=5 and e22629cb[i]>0 and e34428[i]>0 then othpay2[i]=(e22629[i]/e22629cb[i])/(e34428[i]*4.3);
  /* missing value */
    if e22629c[i] in (3,4,5) and -4<e34428[i]<=0 then othpay2[i]=-3;
end;
if e22614[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay2[i]=-3;
if e22614[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay2[i]=e34428[i];

/* non-hourly wage at the beginning, without overtime*/
if e22614[i]=2 and e22629[i]>=0 and e22629e[i]>0 and e34402[i]>0 then othpay2[i]=e22629[i]*e22629e[i]/e34402[i];
if e22614[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e22629[i]>=0 and e34402[i]>0 then
        othpay2[i]=e22629[i]/e34402[i];
if e22614[i]=4 and e22629[i]>=0 and e34402[i]>0 then othpay2[i]=e22629[i]/(2*e34402[i]);
if e22614[i]=5 and e22629[i]>=0 and e34402[i]>0 then othpay2[i]=e22629[i]/(4.3*e34402[i]);
if e22614[i]=6 and e22629[i]>=0 and e34402[i]>0 and e35600[i]>0 then othpay2[i]=e22629[i]/(e35600[i]*e34402[i]);
if e22614[i]=8 and e22629[i]>=0 and e34402[i]>0 then othpay2[i]=e22629[i]/(2.15*e34402[i]);
if e22614[i] in (12,13) and e22629[i]>=0 then do;
   /* condition based on the time unit to finish per othpay2. */
    if e22629c[i]=3 and e22629cb[i]>0 and e22629e[i]>=0 and e34402[i]>0 then
      othpay2[i]=(e22629[i]/e22629cb[i])*e22629e[i]/e34402[i];
    if e22629c[i]=4 and e22629cb[i]>0 and e34402[i]>0 then othpay2[i]=(e22629[i]/e22629cb[i])/e34402[i];
    if e22629c[i]=5 and e22629cb[i]>0 and e34402[i]>0 then othpay2[i]=(e22629[i]/e22629cb[i])/(e34402[i]*4.3);
  /* missing value */
    if e22629c[i] in (3,4,5) and -4<e34402[i]<=0 then othpay2[i]=-3;
end;
if e22614[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay2[i]=-3;
```

```
 if e22614[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
        othpay2[i]=e34402[i];

 if e22614[i]=6 and e35600[i] le 0 then othpay2[i]=-3;
 if e22614[i]=6 and -4<e35600[i]<0 then othpay2[i]=e35600[i];

end;
end;

/********* for bonuses **********/

 /*report hourly wage at the beginning*/
 do i=1 to 9;
 if e216003[i]=1 and e225003[i]>=0 then othpay3[i]=e225003[i];
 if e23901[i]>0 and e216003[i]=2 and e225503[i]>0 and e225003[i]>=0 then
        othpay3[i]=(e225003[i]*e225503[i])/e23901[i];
 if e23901[i]>0 and e216003[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e225003[i]>=0
        then othpay3[i]=e225003[i]/e23901[i];
 if e23901[i]>0 and e216003[i]=4 and e225003[i]>=0 then othpay3[i]=e225003[i]/(2*e23901[i]);
 if e23901[i]>0 and e216003[i]=5 and e225003[i]>=0 then othpay3[i]=e225003[i]/(4.3*e23901[i]);
 if e23901[i]>0 and e216003[i]=6 and e225003[i]>=0 and e35600[i]>0 then
        othpay3[i]=e225003[i]/(e35600[i]*e23901[i]);
 if e23901[i]>0 and e216003[i]=8 and e225003[i]>=0 then othpay3[i]=e225003[i]/(2.15*e23901[i]);
 if e216003[i] in (9,14) then othpay3[i]=0;
 if e216003[i] in (12,13) and e225003[i]>=0 then do;
    /* condition based on the time unit to finish per othpay3. */
     if e225303[i]=1 and e225353[i]>0 then othpay3[i]=(e225003[i]/e225353[i])*60;
     if e225303[i]=2 and e225353[i]>0 then othpay3[i]=e225003[i]/e225353[i];
     if e225303[i]=3 and e225353[i]>0 and e225503[i]>=0 and e23901[i]>0 then
       othpay3[i]=(e225003[i]/e225353[i])*e225503[i]/e23901[i];
     if e225303[i]=4 and e225353[i]>0 and e23901[i]>0 then othpay3[i]=(e225003[i]/e225353[i])/e23901[i];
     if e225303[i]=5 and e225353[i]>0 and e23901[i]>0 then othpay3[i]=(e225003[i]/e225353[i])/(e23901[i]*4.3);
    /* missing value */
     if -4<e225303[i]<0 or -4<e225353[i]<=0 then othpay3[i]=-3;
     if e225303[i] in (3,4,5) and -4<e23901[i]<=0 then othpay3[i]=-3;
     if e225303[i]=3 and e225503[i]<0 then othpay3[i]=-3;
 end;
 if e216003[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e225003[i]<0 then
        othpay3[i]=e225003[i];
 if e216003[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay3[i]=-3;
 if e216003[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay3[i]=e23901[i];
 if e216003[i]=2 and e225503[i] le 0 then othpay3[i]=-3;
 if e216003[i]=2 and -4<e225503[i]<0 then othpay3[i]=e225503[i];
end;

 /* non-hourly wage at the beginning, with overtime*/
 do i=1 to 9;
 if e34428[i]>0 and e216003[i]=2 and e225503[i]>0 and e225003[i]>=0 then
        othpay3[i]=(e225003[i]*e225503[i])/e34428[i];
 if e34428[i]>0 and e216003[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e225003[i]>=0
        then othpay3[i]=e225003[i]/e34428[i];
 if e34428[i]>0 and e216003[i]=4 and e225003[i]>=0 then othpay3[i]=e225003[i]/(2*e34428[i]);
 if e34428[i]>0 and e216003[i]=5 and e225003[i]>=0 then othpay3[i]=e225003[i]/(4.3*e34428[i]);
 if e34428[i]>0 and e216003[i]=6 and e225003[i]>=0 and e35600[i]>0 then
        othpay3[i]=e225003[i]/(e35600[i]*e34428[i]);
 if e34428[i]>0 and e216003[i]=8 and e225003[i]>=0 then othpay3[i]=e225003[i]/(2.15*e34428[i]);
```

```
 if e216003[i] in (12,13) and e225003[i]>=0 then do;
    /* condition based on the time unit to finish per othpay3. */
     if e225303[i]=3 and e225353[i]>0 and e225503[i]>=0 and e34428[i]>0 then
       othpay3[i]=(e225003[i]/e225353[i])*e225503[i]/e34428[i];
     if e225303[i]=4 and e225353[i]>0 and e34428[i]>0 then othpay3[i]=(e225003[i]/e225353[i])/e34428[i];
     if e225303[i]=5 and e225353[i]>0 and e34428[i]>0 then othpay3[i]=(e225003[i]/e225353[i])/(e34428[i]*4.3);
    /* missing value */
     if e225303[i] in (3,4,5) and -4<e34428[i]<=0 then othpay3[i]=-3;
 end;
  if e216003[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay3[i]=-3;
  if e216003[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
          othpay3[i]=e34428[i];
end;

 /* non-hourly wage at the beginning,without overtime*/
 do i=1 to 9;
 if e34402[i]>0 and e216003[i]=2 and e225503[i]>0 and e225003[i]>=0 then
          othpay3[i]=(e225003[i]*e225503[i])/e34402[i];
 if e34402[i]>0 and e216003[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e225003[i]>=0
          then othpay3[i]=e225003[i]/e34402[i];
 if e34402[i]>0 and e216003[i]=4 and e225003[i]>=0 then othpay3[i]=e225003[i]/(2*e34402[i]);
 if e34402[i]>0 and e216003[i]=5 and e225003[i]>=0 then othpay3[i]=e225003[i]/(4.3*e34402[i]);
 if e34402[i]>0 and e216003[i]=6 and e225003[i]>=0 and e35600[i]>0 then
          othpay3[i]=e225003[i]/(e35600[i]*e34402[i]);
 if e34402[i]>0 and e216003[i]=8 and e225003[i]>=0 then othpay3[i]=e225003[i]/(2.15*e34402[i]);
 if e216003[i] in (7,9,12,13,15,16,17,14,18,19,21,22,23,24,25,26,28) then othpf3[i]=othpf3[i]+1;
 if e216003[i] in (12,13) and e225003[i]>=0 then do;
    /* condition based on the time unit to finish per othpay3. */
     if e225303[i]=3 and e225353[i]>0 and e225503[i]>=0 and e34402[i]>0 then
       othpay3[i]=(e225003[i]/e225353[i])*e225503[i]/e34402[i];
     if e225303[i]=4 and e225353[i]>0 and e34402[i]>0 then othpay3[i]=(e225003[i]/e225353[i])/e34402[i];
     if e225303[i]=5 and e225353[i]>0 and e34402[i]>0 then othpay3[i]=(e225003[i]/e225353[i])/(e34402[i]*4.3);
    /* missing value */
     if e225303[i] in (3,4,5) and -4<e34402[i]<=0 then othpay3[i]=-3;
 end;
  if e216003[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay3[i]=-3;
  if e216003[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          othpay3[i]=e34402[i];
 if e216003[i]=6 and e35600[i] le 0 then othpay3[i]=-3;
 if e216003[i]=6 and -4<e35600[i]<0 then othpay3[i]=e35600[i];
 end;

 /* if bonus is corrected in the later part */
 do i=1 to 9;
 if e226047[i]=1 then do;

 /*report hourly wage at the beginning*/
 if e22615[i]=1 and e22630[i]>=0 then othpay3[i]=e22630[i];
 if e22615[i]=2 and e22630[i]>=0 and e22630e[i]>0 and e23901[i]>0 then othpay3[i]=e22630[i]*e22630e[i]/e23901[i];
 if e22615[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e22630[i]>=0 and e23901[i]>0 then
          othpay3[i]=e22630[i]/e23901[i];
 if e22615[i]=4 and e22630[i]>=0 and e23901[i]>0 then othpay3[i]=e22630[i]/(2*e23901[i]);
 if e22615[i]=5 and e22630[i]>=0 and e23901[i]>0 then othpay3[i]=e22630[i]/(4.3*e23901[i]);
 if e22615[i]=6 and e22630[i]>=0 and e23901[i]>0 and e35600[i]>0 then othpay3[i]=e22630[i]/(e35600[i]*e23901[i]);
 if e22615[i]=8 and e22630[i]>=0 and e23901[i]>0 then othpay3[i]=e22630[i]/(2.15*e23901[i]);
  if e22615[i] in (9,14) then othpay3[i]=0;
  if e22615[i] in (12,13) and e22630[i]>=0 then do;
```

```
    /* condition based on the time unit to finish per othpay3. */
     if e22630c[i]=1 and e22630cb[i]>0 then othpay3[i]=(e22630[i]/e22630cb[i])*60;
     if e22630c[i]=2 and e22630cb[i]>0 then othpay3[i]=e22630[i]/e22630cb[i];
     if e22630c[i]=3 and e22630cb[i]>0 and e22630e[i]>=0 and e23901[i]>0 then
       othpay3[i]=(e22630[i]/e22630cb[i])*e22630e[i]/e23901[i];
     if e22630c[i]=4 and e22630cb[i]>0 and e23901[i]>0 then othpay3[i]=(e22630[i]/e22630cb[i])/e23901[i];
     if e22630c[i]=5 and e22630cb[i]>0 and e23901[i]>0 then othpay3[i]=(e22630[i]/e22630cb[i])/(e23901[i]*4.3);
   /* missing value */
     if -4<e22630c[i]<0 or -4<e22630cb[i]<=0 then othpay3[i]=-3;
     if e22630c[i] in (3,4,5) and -4<e23901[i]<=0 then othpay3[i]=-3;
     if e22630c[i]=3 and e22630e[i]<0 then othpay3[i]=-3;
 end;
 if e22615[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e22630[i]<0 then
          othpay3[i]=e22630[i];
 if e22615[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay3[i]=-3;
 if e22615[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
          othpay3[i]=e23901[i];
 if e22615[i]=2 and e22630e[i] le 0 then othpay3[i]=-3;
 if e22615[i]=2 and -4<e22630e[i]<0 then othpay3[i]=e22630e[i];

/* non-hourly wage at the beginning, with overtime*/
if e22615[i]=2 and e22630[i]>=0 and e22630e[i]>0 and e34428[i]>0 then othpay3[i]=e22630[i]*e22630e[i]/e34428[i];
if e22615[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e22630[i]>=0 and e34428[i]>0 then
          othpay3[i]=e22630[i]/e34428[i];
if e22615[i]=4 and e22630[i]>=0 and e34428[i]>0 then othpay3[i]=e22630[i]/(2*e34428[i]);
if e22615[i]=5 and e22630[i]>=0 and e34428[i]>0 then othpay3[i]=e22630[i]/(4.3*e34428[i]);
if e22615[i]=6 and e22630[i]>=0 and e34428[i]>0 and e35600[i]>0 then othpay3[i]=e22630[i]/(e35600[i]*e34428[i]);
if e22615[i]=8 and e22630[i]>=0 and e34428[i]>0 then othpay3[i]=e22630[i]/(2.15*e34428[i]);
 if e22615[i] in (12,13) and e22630[i]>=0 then do;
    /* condition based on the time unit to finish per othpay3. */
     if e22630c[i]=3 and e22630cb[i]>0 and e22630e[i]>=0 and e34428[i]>0 then
       othpay3[i]=(e22630[i]/e22630cb[i])*e22630e[i]/e34428[i];
     if e22630c[i]=4 and e22630cb[i]>0 and e34428[i]>0 then othpay3[i]=(e22630[i]/e22630cb[i])/e34428[i];
     if e22630c[i]=5 and e22630cb[i]>0 and e34428[i]>0 then othpay3[i]=(e22630[i]/e22630cb[i])/(e34428[i]*4.3);
   /* missing value */
     if e22630c[i] in (3,4,5) and -4<e34428[i]<=0 then othpay3[i]=-3;
 end;
 if e22615[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay3[i]=-3;
 if e22615[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
          othpay3[i]=e34428[i];

/* non-hourly wage at the beginning, without overtime*/
if e22615[i]=2 and e22630[i]>=0 and e22630e[i]>0 and e34402[i]>0 then othpay3[i]=e22630[i]*e22630e[i]/e34402[i];
if e22615[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e22630[i]>=0 and e34402[i]>0 then
          othpay3[i]=e22630[i]/e34402[i];
if e22615[i]=4 and e22630[i]>=0 and e34402[i]>0 then othpay3[i]=e22630[i]/(2*e34402[i]);
if e22615[i]=5 and e22630[i]>=0 and e34402[i]>0 then othpay3[i]=e22630[i]/(4.3*e34402[i]);
if e22615[i]=6 and e22630[i]>=0 and e34402[i]>0 and e35600[i]>0 then othpay3[i]=e22630[i]/(e35600[i]*e34402[i]);
if e22615[i]=8 and e22630[i]>=0 and e34402[i]>0 then othpay3[i]=e22630[i]/(2.15*e34402[i]);
if e22615[i] in (12,13) and e22630[i]>=0 then do;
    /* condition based on the time unit to finish per othpay3. */
     if e22630c[i]=3 and e22630cb[i]>0 and e22630e[i]>=0 and e34402[i]>0 then
       othpay3[i]=(e22630[i]/e22630cb[i])*e22630e[i]/e34402[i];
     if e22630c[i]=4 and e22630cb[i]>0 and e34402[i]>0 then othpay3[i]=(e22630[i]/e22630cb[i])/e34402[i];
     if e22630c[i]=5 and e22630cb[i]>0 and e34402[i]>0 then othpay3[i]=(e22630[i]/e22630cb[i])/(e34402[i]*4.3);
   /* missing value */
     if e22630c[i] in (3,4,5) and -4<e34402[i]<=0 then othpay3[i]=-3;
```

```
    end;
    if e22615[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay3[i]=-3;
    if e22615[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
            othpay3[i]=e34402[i];

   if e22615[i]=6 and e35600[i] le 0 then othpay3[i]=-3;
   if e22615[i]=6 and -4<e35600[i]<0 then othpay3[i]=e35600[i];
  end;
  end;

/********** for incentive pay ***********/

   /*report hourly wage at the beginning*/
   do i=1 to 9;
   if e216004[i]=1 and e225004[i]>=0 then othpay4[i]=e225004[i];
   if e23901[i]>0 and e216004[i]=2 and e225504[i]>0 and e225004[i]>=0 then
            othpay4[i]=(e225004[i]*e225504[i])/e23901[i];
   if e23901[i]>0 and e216004[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e225004[i]>=0
            then othpay4[i]=e225004[i]/e23901[i];
   if e23901[i]>0 and e216004[i]=4 and e225004[i]>=0 then othpay4[i]=e225004[i]/(2*e23901[i]);
   if e23901[i]>0 and e216004[i]=5 and e225004[i]>=0 then othpay4[i]=e225004[i]/(4.3*e23901[i]);
   if e23901[i]>0 and e216004[i]=6 and e225004[i]>=0 and e35600[i]>0 then
            othpay4[i]=e225004[i]/(e35600[i]*e23901[i]);
   if e23901[i]>0 and e216004[i]=8 and e225004[i]>=0 then othpay4[i]=e225004[i]/(2.15*e23901[i]);
   if e216004[i] in (9,14) then othpay4[i]=0;
   if e216004[i] in (12,13) and e225004[i]>=0 then do;
      /* condition based on the time unit to finish per othpay4. */
       if e225304[i]=1 and e225354[i]>0 then othpay4[i]=(e225004[i]/e225354[i])*60;
       if e225304[i]=2 and e225354[i]>0 then othpay4[i]=e225004[i]/e225354[i];
       if e225304[i]=3 and e225354[i]>0 and e225504[i]>=0 and e23901[i]>0 then
         othpay4[i]=(e225004[i]/e225354[i])*e225504[i]/e23901[i];
       if e225304[i]=4 and e225354[i]>0 and e23901[i]>0 then othpay4[i]=(e225004[i]/e225354[i])/e23901[i];
       if e225304[i]=5 and e225354[i]>0 and e23901[i]>0 then othpay4[i]=(e225004[i]/e225354[i])/(e23901[i]*4.3);
     /* missing value */
       if -4<e225304[i]<0 or -4<e225354[i]<=0 then othpay4[i]=-3;
       if e225304[i] in (3,4,5) and -4<e23901[i]<=0 then othpay4[i]=-3;
       if e225304[i]=3 and e225504[i]<0 then othpay4[i]=-3;
   end;
    if e216004[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e225004[i]<0 then
            othpay4[i]=e225004[i];
    if e216004[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay4[i]=-3;
    if e216004[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
            othpay4[i]=e23901[i];
    if e216004[i]=2 and e225504[i] le 0 then othpay4[i]=-3;
    if e216004[i]=2 and -4<e225504[i]<0 then othpay4[i]=e225504[i];

   end;

   /* non-hourly wage at the beginning, with overtime*/
   do i=1 to 9;
   if e34428[i]>0 and e216004[i]=2 and e225504[i]>0 and e225004[i]>=0 then
            othpay4[i]=(e225004[i]*e225504[i])/e34428[i];
   if e34428[i]>0 and e216004[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e225004[i]>=0
            then othpay4[i]=e225004[i]/e34428[i];
   if e34428[i]>0 and e216004[i]=4 and e225004[i]>=0 then othpay4[i]=e225004[i]/(2*e34428[i]);
   if e34428[i]>0 and e216004[i]=5 and e225004[i]>=0 then othpay4[i]=e225004[i]/(4.3*e34428[i]);
```

```
if e34428[i]>0 and e216004[i]=6 and e225004[i]>=0 and e35600[i]>0 then
        othpay4[i]=e225004[i]/(e35600[i]*e34428[i]);
if e34428[i]>0 and e216004[i]=8 and e225004[i]>=0 then othpay4[i]=e225004[i]/(2.15*e34428[i]);
if e216004[i] in (12,13) and e225004[i]>=0 then do;
   /* condition based on the time unit to finish per othpay4. */
    if e225304[i]=3 and e225354[i]>0 and e225504[i]>=0 and e34428[i]>0 then
      othpay4[i]=(e225004[i]/e225354[i])*e225504[i]/e34428[i];
    if e225304[i]=4 and e225354[i]>0 and e34428[i]>0 then othpay4[i]=(e225004[i]/e225354[i])/e34428[i];
    if e225304[i]=5 and e225354[i]>0 and e34428[i]>0 then othpay4[i]=(e225004[i]/e225354[i])/(e34428[i]*4.3);
  /* missing value */
    if e225304[i] in (3,4,5) and -4<e34428[i]<=0 then othpay4[i]=-3;
end;
 if e216004[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay4[i]=-3;
 if e216004[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay4[i]=e34428[i];
end;

/* non-hourly wage at the beginning, without overtime*/
do i=1 to 9;
if e34402[i]>0 and e216004[i]=2 and e225504[i]>0 and e225004[i]>=0 then
        othpay4[i]=(e225004[i]*e225504[i])/e34402[i];
if e34402[i]>0 and e216004[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e225004[i]>=0
        then othpay4[i]=e225004[i]/e34402[i];
if e34402[i]>0 and e216004[i]=4 and e225004[i]>=0 then othpay4[i]=e225004[i]/(2*e34402[i]);
if e34402[i]>0 and e216004[i]=5 and e225004[i]>=0 then othpay4[i]=e225004[i]/(4.3*e34402[i]);
if e34402[i]>0 and e216004[i]=6 and e225004[i]>=0 and e35600[i]>0 then
        othpay4[i]=e225004[i]/(e35600[i]*e34402[i]);
if e34402[i]>0 and e216004[i]=8 and e225004[i]>=0 then othpay4[i]=e225004[i]/(2.15*e34402[i]);
if e216004[i] in (7,9,12,13,15,16,17,14,18,19,21,22,23,24,25,26,28) then othpf4[i]=othpf4[i]+1;
if e216004[i] in (12,13) and e225004[i]>=0 then do;
   /* condition based on the time unit to finish per othpay4. */
    if e225304[i]=3 and e225354[i]>0 and e225504[i]>=0 and e34402[i]>0 then
      othpay4[i]=(e225004[i]/e225354[i])*e225504[i]/e34402[i];
    if e225304[i]=4 and e225354[i]>0 and e34402[i]>0 then othpay4[i]=(e225004[i]/e225354[i])/e34402[i];
    if e225304[i]=5 and e225354[i]>0 and e34402[i]>0 then othpay4[i]=(e225004[i]/e225354[i])/(e34402[i]*4.3);
  /* missing value */
    if e225304[i] in (3,4,5) and -4<e34402[i]<=0 then othpay4[i]=-3;
end;
 if e216004[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay4[i]=-3;
 if e216004[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
        othpay4[i]=e34402[i];

if e216004[i]=6 and e35600[i] le 0 then othpay4[i]=-3;
if e216004[i]=6 and -4<e35600[i]<0 then othpay4[i]=e35600[i];

end;

/* if incentive pay is corrected in the later part */
do i=1 to 9;
if e226048[i]=1 then do;

/*report hourly wage at the beginning*/
if e22616[i]=1 and e22631[i]>=0 then othpay4[i]=e22631[i];
if e22616[i]=2 and e22631[i]>=0 and e22631e[i]>0 and e23901[i]>0 then othpay4[i]=e22631[i]*e22631e[i]/e23901[i];
if e22616[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e22631[i]>=0 and e23901[i]>0 then
        othpay4[i]=e22631[i]/e23901[i];
if e22616[i]=4 and e22631[i]>=0 and e23901[i]>0 then othpay4[i]=e22631[i]/(2*e23901[i]);
```

```
if e22616[i]=5 and e22631[i]>=0 and e23901[i]>0 then othpay4[i]=e22631[i]/(4.3*e23901[i]);
if e22616[i]=6 and e22631[i]>=0 and e23901[i]>0 and e35600[i]>0 then othpay4[i]=e22631[i]/(e35600[i]*e23901[i]);
if e22616[i]=8 and e22631[i]>=0 and e23901[i]>0 then othpay4[i]=e22631[i]/(2.15*e23901[i]);
if e22616[i] in (9,14) then othpay4[i]=0;
if e22616[i] in (12,13) and e22631[i]>=0 then do;
   /* condition based on the time unit to finish per othpay4. */
    if e22631c[i]=1 and e22631cb[i]>0 then othpay4[i]=(e22631[i]/e22631cb[i])*60;
    if e22631c[i]=2 and e22631cb[i]>0 then othpay4[i]=e22631[i]/e22631cb[i];
    if e22631c[i]=3 and e22631cb[i]>0 and e22631e[i]>=0 and e23901[i]>0 then
      othpay4[i]=(e22631[i]/e22631cb[i])*e22631e[i]/e23901[i];
    if e22631c[i]=4 and e22631cb[i]>0 and e23901[i]>0 then othpay4[i]=(e22631[i]/e22631cb[i])/e23901[i];
    if e22631c[i]=5 and e22631cb[i]>0 and e23901[i]>0 then othpay4[i]=(e22631[i]/e22631cb[i])/(e23901[i]*4.3);
  /* missing value */
    if -4<e22631c[i]<0 or -4<e22631cb[i]<=0 then othpay4[i]=-3;
    if e22631c[i] in (3,4,5) and -4<e23901[i]<=0 then othpay4[i]=-3;
    if e22631c[i]=3 and e22631e[i]<0 then othpay4[i]=-3;
end;
if e22616[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e22631[i]<0 then
        othpay4[i]=e22631[i];
if e22616[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay4[i]=-3;
if e22616[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay4[i]=e23901[i];
if e22616[i]=2 and e22631e[i] le 0 then othpay4[i]=-3;
if e22616[i]=2 and -4<e22631e[i]<0 then othpay4[i]=e22631e[i];

/* non-hourly wage at the beginning, with overtime*/
if e22616[i]=2 and e22631[i]>=0 and e22631e[i]>0 and e34428[i]>0 then othpay4[i]=e22631[i]*e22631e[i]/e34428[i];
if e22616[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e22631[i]>=0 and e34428[i]>0 then
        othpay4[i]=e22631[i]/e34428[i];
if e22616[i]=4 and e22631[i]>=0 and e34428[i]>0 then othpay4[i]=e22631[i]/(2*e34428[i]);
if e22616[i]=5 and e22631[i]>=0 and e34428[i]>0 then othpay4[i]=e22631[i]/(4.3*e34428[i]);
if e22616[i]=6 and e22631[i]>=0 and e34428[i]>0 and e35600[i]>0 then othpay4[i]=e22631[i]/(e35600[i]*e34428[i]);
if e22616[i]=8 and e22631[i]>=0 and e34428[i]>0 then othpay4[i]=e22631[i]/(2.15*e34428[i]);
if e22616[i] in (12,13) and e22631[i]>=0 then do;
   /* condition based on the time unit to finish per othpay4. */
    if e22631c[i]=3 and e22631cb[i]>0 and e22631e[i]>=0 and e34428[i]>0 then
      othpay4[i]=(e22631[i]/e22631cb[i])*e22631e[i]/e34428[i];
    if e22631c[i]=4 and e22631cb[i]>0 and e34428[i]>0 then othpay4[i]=(e22631[i]/e22631cb[i])/e34428[i];
    if e22631c[i]=5 and e22631cb[i]>0 and e34428[i]>0 then othpay4[i]=(e22631[i]/e22631cb[i])/(e34428[i]*4.3);
  /* missing value */
    if e22631c[i] in (3,4,5) and -4<e34428[i]<=0 then othpay4[i]=-3;
end;
if e22616[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay4[i]=-3;
if e22616[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay4[i]=e34428[i];

/* non-hourly wage at the beginning, without overtime*/
if e22616[i]=2 and e22631[i]>=0 and e22631e[i]>0 and e34402[i]>0 then othpay4[i]=e22631[i]*e22631e[i]/e34402[i];
if e22616[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e22631[i]>=0 and e34402[i]>0 then
        othpay4[i]=e22631[i]/e34402[i];
if e22616[i]=4 and e22631[i]>=0 and e34402[i]>0 then othpay4[i]=e22631[i]/(2*e34402[i]);
if e22616[i]=5 and e22631[i]>=0 and e34402[i]>0 then othpay4[i]=e22631[i]/(4.3*e34402[i]);
if e22616[i]=6 and e22631[i]>=0 and e34402[i]>0 and e35600[i]>0 then othpay4[i]=e22631[i]/(e35600[i]*e34402[i]);
if e22616[i]=8 and e22631[i]>=0 and e34402[i]>0 then othpay4[i]=e22631[i]/(2.15*e34402[i]);
if e22616[i] in (12,13) and e22631[i]>=0 then do;
   /* condition based on the time unit to finish per othpay4. */
    if e22631c[i]=3 and e22631cb[i]>0 and e22631e[i]>=0 and e34402[i]>0 then
```

```
        othpay4[i]=(e22631[i]/e22631cb[i])*e22631e[i]/e34402[i];
      if e22631c[i]=4 and e22631cb[i]>0 and e34402[i]>0 then othpay4[i]=(e22631[i]/e22631cb[i])/e34402[i];
      if e22631c[i]=5 and e22631cb[i]>0 and e34402[i]>0 then othpay4[i]=(e22631[i]/e22631cb[i])/(e34402[i]*4.3);
    /* missing value */
      if e22631c[i] in (3,4,5) and -4<e34402[i]<=0 then othpay4[i]=-3;
   end;
   if e22616[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay4[i]=-3;
   if e22616[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          othpay4[i]=e34402[i];

  if e22616[i]=6 and e35600[i] le 0 then othpay4[i]=-3;
  if e22616[i]=6 and -4<e35600[i]<0 then othpay4[i]=e35600[i];
 end;
 end;

/********* for others *********/

  /*report hourly wage at the beginning*/
  do i=1 to 9;
  if e216005[i]=1 and e225005[i]>=0 then othpay5[i]=e225005[i];
  if e23901[i]>0 and e216005[i]=2 and e225505[i]>0 and e225005[i]>=0 then
          othpay5[i]=(e225005[i]*e225505[i])/e23901[i];
  if e23901[i]>0 and e216005[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e225005[i]>=0
          then othpay5[i]=e225005[i]/e23901[i];
  if e23901[i]>0 and e216005[i]=4 and e225005[i]>=0 then othpay5[i]=e225005[i]/(2*e23901[i]);
  if e23901[i]>0 and e216005[i]=5 and e225005[i]>=0 then othpay5[i]=e225005[i]/(4.3*e23901[i]);
  if e23901[i]>0 and e216005[i]=6 and e225005[i]>=0 and e35600[i]>0 then
          othpay5[i]=e225005[i]/(e35600[i]*e23901[i]);
  if e23901[i]>0 and e216005[i]=8 and e225005[i]>=0 then othpay5[i]=e225005[i]/(2.15*e23901[i]);
  if e216005[i] in (9,14) then othpay5[i]=0;
  if e216005[i] in (12,13) and e225005[i]>=0 then do;
     /* condition based on the time unit to finish per othpay5. */
      if e225305[i]=1 and e225355[i]>0 then othpay5[i]=(e225005[i]/e225355[i])*60;
      if e225305[i]=2 and e225355[i]>0 then othpay5[i]=e225005[i]/e225355[i];
      if e225305[i]=3 and e225355[i]>0 and e225505[i]>=0 and e23901[i]>0 then
        othpay5[i]=(e225005[i]/e225355[i])*e225505[i]/e23901[i];
      if e225305[i]=4 and e225355[i]>0 and e23901[i]>0 then othpay5[i]=(e225005[i]/e225355[i])/e23901[i];
      if e225305[i]=5 and e225355[i]>0 and e23901[i]>0 then othpay5[i]=(e225005[i]/e225355[i])/(e23901[i]*4.3);
    /* missing value */
      if -4<e225305[i]<0 or -4<e225355[i]<=0 then othpay5[i]=-3;
      if e225305[i] in (3,4,5) and -4<e23901[i]<=0 then othpay5[i]=-3;
      if e225305[i]=3 and e225505[i]<0 then othpay5[i]=-3;
  end;
   if e216005[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e225005[i]<0 then
          othpay5[i]=e225005[i];
   if e216005[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay5[i]=-3;
   if e216005[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
          othpay5[i]=e23901[i];
   if e216005[i]=2 and e225505[i] le 0 then othpay5[i]=-3;
   if e216005[i]=2 and -4<e225505[i]<0 then othpay5[i]=e225505[i];
 end;

  /* non-hourly wage at the beginning, with overtime*/
  do i=1 to 9;
  if e34428[i]>0 and e216005[i]=2 and e225505[i]>0 and e225005[i]>=0 then
          othpay5[i]=(e225005[i]*e225505[i])/e34428[i];
```

```
if e34428[i]>0 and e216005[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e225005[i]>=0
        then othpay5[i]=e225005[i]/e34428[i];
if e34428[i]>0 and e216005[i]=4 and e225005[i]>=0 then othpay5[i]=e225005[i]/(2*e34428[i]);
if e34428[i]>0 and e216005[i]=5 and e225005[i]>=0 then othpay5[i]=e225005[i]/(4.3*e34428[i]);
if e34428[i]>0 and e216005[i]=6 and e225005[i]>=0 and e35600[i]>0 then
        othpay5[i]=e225005[i]/(e35600[i]*e34428[i]);
if e34428[i]>0 and e216005[i]=8 and e225005[i]>=0 then othpay5[i]=e225005[i]/(2.15*e34428[i]);
 if e216005[i] in (12,13) and e225005[i]>=0 then do;
   /* condition based on the time unit to finish per othpay5. */
    if e225305[i]=3 and e225355[i]>0 and e225505[i]>=0 and e34428[i]>0 then
      othpay5[i]=(e225005[i]/e225355[i])*e225505[i]/e34428[i];
    if e225305[i]=4 and e225355[i]>0 and e34428[i]>0 then othpay5[i]=(e225005[i]/e225355[i])/e34428[i];
    if e225305[i]=5 and e225355[i]>0 and e34428[i]>0 then othpay5[i]=(e225005[i]/e225355[i])/(e34428[i]*4.3);
  /* missing value */
    if e225305[i] in (3,4,5) and -4<e34428[i]<=0 then othpay5[i]=-3;
end;
 if e216005[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay5[i]=-3;
 if e216005[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay5[i]=e34428[i];

end;

/* non-hourly wage at the beginning, without overtime*/
do i=1 to 9;
if e34402[i]>0 and e216005[i]=2 and e225505[i]>0 and e225005[i]>=0 then
        othpay5[i]=(e225005[i]*e225505[i])/e34402[i];
if e34402[i]>0 and e216005[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e225005[i]>=0
        then othpay5[i]=e225005[i]/e34402[i];
if e34402[i]>0 and e216005[i]=4 and e225005[i]>=0 then othpay5[i]=e225005[i]/(2*e34402[i]);
if e34402[i]>0 and e216005[i]=5 and e225005[i]>=0 then othpay5[i]=e225005[i]/(4.3*e34402[i]);
if e34402[i]>0 and e216005[i]=6 and e225005[i]>=0 and e35600[i]>0 then
        othpay5[i]=e225005[i]/(e35600[i]*e34402[i]);
if e34402[i]>0 and e216005[i]=8 and e225005[i]>=0 then othpay5[i]=e225005[i]/(2.15*e34402[i]);
if e216005[i] in (7,9,12,13,15,16,17,14,18,19,21,22,23,24,25,26,28) then othpf5[i]=othpf5[i]+1;
 if e216005[i] in (12,13) and e225005[i]>=0 then do;
   /* condition based on the time unit to finish per othpay5. */
    if e225305[i]=3 and e225355[i]>0 and e225505[i]>=0 and e34402[i]>0 then
      othpay5[i]=(e225005[i]/e225355[i])*e225505[i]/e34402[i];
    if e225305[i]=4 and e225355[i]>0 and e34402[i]>0 then othpay5[i]=(e225005[i]/e225355[i])/e34402[i];
    if e225305[i]=5 and e225355[i]>0 and e34402[i]>0 then othpay5[i]=(e225005[i]/e225355[i])/(e34402[i]*4.3);
  /* missing value */
    if e225305[i] in (3,4,5) and -4<e34402[i]<=0 then othpay5[i]=-3;
end;
 if e216005[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay5[i]=-3;
 if e216005[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
        othpay5[i]=e34402[i];

if e216005[i]=6 and e35600[i] le 0 then othpay5[i]=-3;
if e216005[i]=6 and -4<e35600[i]<0 then othpay5[i]=e35600[i];

end;

/* if other compensation is corrected in the later part */
do i=1 to 9;
if e226049[i]=1 then do;

/*report hourly wage at the beginnnig*/
```

```
if e22617[i]=1 and e22632[i]>=0 then othpay5[i]=e22632[i];
if e22617[i]=2 and e22632[i]>=0 and e22632e[i]>0 and e23901[i]>0 then othpay5[i]=e22632[i]*e22632e[i]/e23901[i];
if e22617[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e22632[i]>=0 and e23901[i]>0 then
        othpay5[i]=e22632[i]/e23901[i];
if e22617[i]=4 and e22632[i]>=0 and e23901[i]>0 then othpay5[i]=e22632[i]/(2*e23901[i]);
if e22617[i]=5 and e22632[i]>=0 and e23901[i]>0 then othpay5[i]=e22632[i]/(4.3*e23901[i]);
if e22617[i]=6 and e22632[i]>=0 and e23901[i]>0 and e35600[i]>0 then othpay5[i]=e22632[i]/(e35600[i]*e23901[i]);
if e22617[i]=8 and e22632[i]>=0 and e23901[i]>0 then othpay5[i]=e22632[i]/(2.15*e23901[i]);
if e22617[i] in (9,14) then othpay5[i]=0;
 if e22617[i] in (12,13) and e22632[i]>=0 then do;
   /* condition based on the time unit to finish per othpay5. */
    if e22632c[i]=1 and e22632cb[i]>0 then othpay5[i]=(e22632[i]/e22632cb[i])*60;
    if e22632c[i]=2 and e22632cb[i]>0 then othpay5[i]=e22632[i]/e22632cb[i];
    if e22632c[i]=3 and e22632cb[i]>0 and e22632e[i]>=0 and e23901[i]>0 then
      othpay5[i]=(e22632[i]/e22632cb[i])*e22632e[i]/e23901[i];
    if e22632c[i]=4 and e22632cb[i]>0 and e23901[i]>0 then othpay5[i]=(e22632[i]/e22632cb[i])/e23901[i];
    if e22632c[i]=5 and e22632cb[i]>0 and e23901[i]>0 then othpay5[i]=(e22632[i]/e22632cb[i])/(e23901[i]*4.3);
  /* missing value */
    if -4<e22632c[i]<0 or -4<e22632cb[i]<=0 then othpay5[i]=-3;
    if e22632c[i] in (3,4,5) and -4<e23901[i]<=0 then othpay5[i]=-3;
    if e22632c[i]=3 and e22632e[i]<0 then othpay5[i]=-3;
 end;
 if e22617[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e22632[i]<0 then
        othpay5[i]=e22632[i];
 if e22617[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay5[i]=-3;
 if e22617[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay5[i]=e23901[i];
 if e22617[i]=2 and e22632e[i] le 0 then othpay5[i]=-3;
 if e22617[i]=2 and -4<e22632e[i]<0 then othpay5[i]=e22632e[i];

/* non-hourly wage at the beginning, with overtime*/
if e22617[i]=2 and e22632[i]>=0 and e22632e[i]>0 and e34428[i]>0 then othpay5[i]=e22632[i]*e22632e[i]/e34428[i];
if e22617[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e22632[i]>=0 and e34428[i]>0 then
        othpay5[i]=e22632[i]/e34428[i];
if e22617[i]=4 and e22632[i]>=0 and e34428[i]>0 then othpay5[i]=e22632[i]/(2*e34428[i]);
if e22617[i]=5 and e22632[i]>=0 and e34428[i]>0 then othpay5[i]=e22632[i]/(4.3*e34428[i]);
if e22617[i]=6 and e22632[i]>=0 and e34428[i]>0 and e35600[i]>0 then othpay5[i]=e22632[i]/(e35600[i]*e34428[i]);
if e22617[i]=8 and e22632[i]>=0 and e34428[i]>0 then othpay5[i]=e22632[i]/(2.15*e34428[i]);
if e22617[i] in (12,13) and e22632[i]>=0 then do;
   /* condition based on the time unit to finish per othpay5. */
    if e22632c[i]=3 and e22632cb[i]>0 and e22632e[i]>=0 and e34428[i]>0 then
      othpay5[i]=(e22632[i]/e22632cb[i])*e22632e[i]/e34428[i];
    if e22632c[i]=4 and e22632cb[i]>0 and e34428[i]>0 then othpay5[i]=(e22632[i]/e22632cb[i])/e34428[i];
    if e22632c[i]=5 and e22632cb[i]>0 and e34428[i]>0 then othpay5[i]=(e22632[i]/e22632cb[i])/(e34428[i]*4.3);
  /* missing value */
    if e22632c[i] in (3,4,5) and -4<e34428[i]<=0 then othpay5[i]=-3;
 end;
 if e22617[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay5[i]=-3;
 if e22617[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay5[i]=e34428[i];

/* non-hourly wage at the beginning, without overtime*/
if e22617[i]=2 and e22632[i]>=0 and e22632e[i]>0 and e34402[i]>0 then othpay5[i]=e22632[i]*e22632e[i]/e34402[i];
if e22617[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e22632[i]>=0 and e34402[i]>0 then
        othpay5[i]=e22632[i]/e34402[i];
if e22617[i]=4 and e22632[i]>=0 and e34402[i]>0 then othpay5[i]=e22632[i]/(2*e34402[i]);
if e22617[i]=5 and e22632[i]>=0 and e34402[i]>0 then othpay5[i]=e22632[i]/(4.3*e34402[i]);
```

```
  if e22617[i]=6 and e22632[i]>=0 and e34402[i]>0 and e35600[i]>0 then othpay5[i]=e22632[i]/(e35600[i]*e34402[i]);
  if e22617[i]=8 and e22632[i]>=0 and e34402[i]>0 then othpay5[i]=e22632[i]/(2.15*e34402[i]);
  if e22617[i] in (12,13) and e22632[i]>=0 then do;
     /* condition based on the time unit to finish per othpay5. */
      if e22632c[i]=3 and e22632cb[i]>0 and e22632e[i]>=0 and e34402[i]>0 then
        othpay5[i]=(e22632[i]/e22632cb[i])*e22632e[i]/e34402[i];
      if e22632c[i]=4 and e22632cb[i]>0 and e34402[i]>0 then othpay5[i]=(e22632[i]/e22632cb[i])/e34402[i];
      if e22632c[i]=5 and e22632cb[i]>0 and e34402[i]>0 then othpay5[i]=(e22632[i]/e22632cb[i])/(e34402[i]*4.3);
    /* missing value */
      if e22632c[i] in (3,4,5) and -4<e34402[i]<=0 then othpay5[i]=-3;
   end;
   if e22617[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay5[i]=-3;
   if e22617[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          othpay5[i]=e34402[i];
  if e22617[i]=6 and e35600[i] le 0 then othpay5[i]=-3;
  if e22617[i]=6 and -4<e35600[i]<0 then othpay5[i]=e35600[i];
 end;
 end;

/******** do it for non-overtime payment if answered in the later part *********/
/******** for tips ********/

  /*report hourly wage at the beginning*/
  do i=1 to 9;
  if e102051[i]=1 and e102141[i]>=0 then othpay1[i]=e102141[i];
  if e88000[i]>0 and e102051[i]=2 and e10214b1[i]>0 and e102141[i]>=0 then
          othpay1[i]=(e102141[i]*e10214b1[i])/e88000[i];
  if e88000[i]>0 and e102051[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e102141[i]>=0
          then othpay1[i]=e102141[i]/e88000[i];
  if e88000[i]>0 and e102051[i]=4 and e102141[i]>=0 then othpay1[i]=e102141[i]/(2*e88000[i]);
  if e88000[i]>0 and e102051[i]=5 and e102141[i]>=0 then othpay1[i]=e102141[i]/(4.3*e88000[i]);
  if e88000[i]>0 and e102051[i]=6 and e102141[i]>=0 and e99500[i]>0 then
          othpay1[i]=e102141[i]/(e99500[i]*e88000[i]);
  if e88000[i]>0 and e102051[i]=8 and e102141[i]>=0 then othpay1[i]=e102141[i]/(2.15*e88000[i]);
  if e102051[i] in (9,14) then othpay1[i]=0;
  if e102051[i] in (12,13) and e102141[i]>=0 then do;
     /* condition based on the time unit to finish per othpay1. */
      if e10214aaa1[i]=1 and e10214aab1[i]>0 then othpay1[i]=(e102141[i]/e10214aab1[i])*60;
      if e10214aaa1[i]=2 and e10214aab1[i]>0 then othpay1[i]=e102141[i]/e10214aab1[i];
      if e10214aaa1[i]=3 and e10214aab1[i]>0 and e10214b1[i]>=0 and e88000[i]>0 then
        othpay1[i]=(e102141[i]/e10214aab1[i])*e10214b1[i]/e88000[i];
      if e10214aaa1[i]=4 and e10214aab1[i]>0 and e88000[i]>0 then othpay1[i]=(e102141[i]/e10214aab1[i])/e88000[i];
      if e10214aaa1[i]=5 and e10214aab1[i]>0 and e88000[i]>0 then
          othpay1[i]=(e102141[i]/e10214aab1[i])/(e88000[i]*4.3);
    /* missing value */
      if -4<e10214aaa1[i]<0 or -4<e10214aab1[i]<=0 then othpay1[i]=-3;
      if e10214aaa1[i] in (3,4,5) and -4<e88000[i]<=0 then othpay1[i]=-3;
      if e10214aaa1[i]=3 and e10214b1[i]<0 then othpay1[i]=-3;
  end;
   if e102051[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e102141[i]<0 then
          othpay1[i]=e102141[i];
   if e102051[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e88000[i]=0 then othpay1[i]=-3;
   if e102051[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e88000[i]<0 then
          othpay1[i]=e88000[i];
   if e102051[i]=2 and e10214b1[i] le 0 then othpay1[i]=-3;
   if e102051[i]=2 and -4<e10214b1[i]<0 then othpay1[i]=e10214b1[i];
```

```
end;

/* non-hourly wage at the beginning, with overtime*/
do i=1 to 9;
if e98429[i]>0 and e102051[i]=2 and e10214b1[i]>0 and e102141[i]>=0 then
        othpay1[i]=(e102141[i]*e10214b1[i])/e98429[i];
if e98429[i]>0 and e102051[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e102141[i]>=0
        then othpay1[i]=e102141[i]/e98429[i];
if e98429[i]>0 and e102051[i]=4 and e102141[i]>=0 then othpay1[i]=e102141[i]/(2*e98429[i]);
if e98429[i]>0 and e102051[i]=5 and e102141[i]>=0 then othpay1[i]=e102141[i]/(4.3*e98429[i]);
if e98429[i]>0 and e102051[i]=6 and e102141[i]>=0 and e99500[i]>0 then
        othpay1[i]=e102141[i]/(e99500[i]*e98429[i]);
if e98429[i]>0 and e102051[i]=8 and e102141[i]>=0 then othpay1[i]=e102141[i]/(2.15*e98429[i]);
if e102051[i] in (12,13) and e102141[i]>=0 then do;
   /* condition based on the time unit to finish per othpay1. */
    if e10214aaa1[i]=3 and e10214aab1[i]>0 and e10214b1[i]>=0 and e98429[i]>0 then
      othpay1[i]=(e102141[i]/e10214aab1[i])*e10214b1[i]/e98429[i];
    if e10214aaa1[i]=4 and e10214aab1[i]>0 and e98429[i]>0 then othpay1[i]=(e102141[i]/e10214aab1[i])/e98429[i];
    if e10214aaa1[i]=5 and e10214aab1[i]>0 and e98429[i]>0 then
        othpay1[i]=(e102141[i]/e10214aab1[i])/(e98429[i]*4.3);
  /* missing value */
    if e10214aaa1[i] in (3,4,5) and -4<e98429[i]<=0 then othpay1[i]=-3;
end;
 if e102051[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98429[i]=0 then othpay1[i]=-3;
 if e102051[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98429[i]<0 then
        othpay1[i]=e98429[i];

end;

/* non-hourly wage at the beginning, without overtime*/
do i=1 to 9;
if e98402[i]>0 and e102051[i]=2 and e10214b1[i]>0 and e102141[i]>=0 then
        othpay1[i]=(e102141[i]*e10214b1[i])/e98402[i];
if e98402[i]>0 and e102051[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e102141[i]>=0
        then othpay1[i]=e102141[i]/e98402[i];
if e98402[i]>0 and e102051[i]=4 and e102141[i]>=0 then othpay1[i]=e102141[i]/(2*e98402[i]);
if e98402[i]>0 and e102051[i]=5 and e102141[i]>=0 then othpay1[i]=e102141[i]/(4.3*e98402[i]);
if e98402[i]>0 and e102051[i]=6 and e102141[i]>=0 and e99500[i]>0 then othpay1[i]=e102141[i]/(52*e98402[i]);
if e98402[i]>0 and e102051[i]=8 and e102141[i]>=0 then othpay1[i]=e102141[i]/(2.15*e98402[i]);
if e102051[i] in (7,9,12,13,15,16,17,14,18,19,21,22,23,24,25,26,28) then othpf1[i]=othpf1[i]+1;
if e102051[i] in (12,13) and e102141[i]>=0 then do;
   /* condition based on the time unit to finish per othpay1. */
    if e10214aaa1[i]=3 and e10214aab1[i]>0 and e10214b1[i]>=0 and e98402[i]>0 then
      othpay1[i]=(e102141[i]/e10214aab1[i])*e10214b1[i]/e98402[i];
    if e10214aaa1[i]=4 and e10214aab1[i]>0 and e98402[i]>0 then othpay1[i]=(e102141[i]/e10214aab1[i])/e98402[i];
    if e10214aaa1[i]=5 and e10214aab1[i]>0 and e98402[i]>0 then
        othpay1[i]=(e102141[i]/e10214aab1[i])/(e98402[i]*4.3);
  /* missing value */
    if e10214aaa1[i] in (3,4,5) and -4<e98402[i]<=0 then othpay1[i]=-3;
end;
 if e102051[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98402[i]=0 then othpay1[i]=-3;
 if e102051[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98402[i]<0 then
        othpay1[i]=e98402[i];
if e102051[i]=6 and e99500[i] le 0 then othpay1[i]=-3;
if e102051[i]=6 and -4<e99500[i]<0 then othpay1[i]=e99500[i];
end;
```

```
/* if tips is corrected in the later part */
  do i=1 to 9;
  if e102255[i]=1 then do;

  /*report hourly wage at the beginning*/
  if e100234[i]=1 and e100250[i]>=0 then othpay1[i]=e100250[i];
  if e100234[i]=2 and e100250[i]>=0 and e100250e[i]>0 and e88000[i]>0 then
          othpay1[i]=e100250[i]*e100250e[i]/e88000[i];
  if e100234[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e100250[i]>=0 and e88000[i]>0
          then othpay1[i]=e100250[i]/e88000[i];
  if e100234[i]=4 and e100250[i]>=0 and e88000[i]>0 then othpay1[i]=e100250[i]/(2*e88000[i]);
  if e100234[i]=5 and e100250[i]>=0 and e88000[i]>0 then othpay1[i]=e100250[i]/(4.3*e88000[i]);
  if e100234[i]=6 and e100250[i]>=0 and e88000[i]>0 and e99500[i]>0 then
          othpay1[i]=e100250[i]/(e99500[i]*e88000[i]);
  if e100234[i]=8 and e100250[i]>=0 and e88000[i]>0 then othpay1[i]=e100250[i]/(2.15*e88000[i]);
  if e100234[i] in (9,14) then othpay1[i]=0;
  if e100234[i] in (12,13) and e100250[i]>=0 then do;
     /* condition based on the time unit to finish per othpay1. */
      if e100250c[i]=1 and e100250cb[i]>0 then othpay1[i]=(e100250[i]/e100250cb[i])*60;
      if e100250c[i]=2 and e100250cb[i]>0 then othpay1[i]=e100250[i]/e100250cb[i];
      if e100250c[i]=3 and e100250cb[i]>0 and e100250e[i]>=0 and e88000[i]>0 then
        othpay1[i]=(e100250[i]/e100250cb[i])*e100250e[i]/e88000[i];
      if e100250c[i]=4 and e100250cb[i]>0 and e88000[i]>0 then othpay1[i]=(e100250[i]/e100250cb[i])/e88000[i];
      if e100250c[i]=5 and e100250cb[i]>0 and e88000[i]>0 then
          othpay1[i]=(e100250[i]/e100250cb[i])/(e88000[i]*4.3);
    /* missing value */
      if -4<e100250c[i]<0 or -4<e100250cb[i]<=0 then othpay1[i]=-3;
      if e100250c[i] in (3,4,5) and -4<e88000[i]<=0 then othpay1[i]=-3;
      if e100250c[i]=3 and e100250e[i]<0 then othpay1[i]=-3;
  end;
   if e100234[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e100250[i]<0 then
          othpay1[i]=e100250[i];
   if e100234[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e88000[i]=0 then othpay1[i]=-3;
   if e100234[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e88000[i]<0 then
          othpay1[i]=e88000[i];
   if e100234[i]=2 and e100250e[i] le 0 then othpay1[i]=-3;
   if e100234[i]=2 and -4<e100250e[i]<0 then othpay1[i]=e100250e[i];

  /* non-hourly wage at the beginning, with overtime*/
  if e100234[i]=2 and e100250[i]>=0 and e100250e[i]>0 and e98429[i]>0 then
          othpay1[i]=e100250[i]*e100250e[i]/e98429[i];
  if e100234[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e100250[i]>=0 and e98429[i]>0
          then othpay1[i]=e100250[i]/e98429[i];
  if e100234[i]=4 and e100250[i]>=0 and e98429[i]>0 then othpay1[i]=e100250[i]/(2*e98429[i]);
  if e100234[i]=5 and e100250[i]>=0 and e98429[i]>0 then othpay1[i]=e100250[i]/(4.3*e98429[i]);
  if e100234[i]=6 and e100250[i]>=0 and e98429[i]>0 and e99500[i]>0 then
          othpay1[i]=e100250[i]/(e99500[i]*e98429[i]);
  if e100234[i]=8 and e100250[i]>=0 and e98429[i]>0 then othpay1[i]=e100250[i]/(2.15*e98429[i]);
  if e100234[i] in (12,13) and e100250[i]>=0 then do;
     /* condition based on the time unit to finish per othpay1. */
      if e100250c[i]=3 and e100250cb[i]>0 and e100250e[i]>=0 and e98429[i]>0 then
        othpay1[i]=(e100250[i]/e100250cb[i])*e100250e[i]/e98429[i];
      if e100250c[i]=4 and e100250cb[i]>0 and e98429[i]>0 then othpay1[i]=(e100250[i]/e100250cb[i])/e98429[i];
      if e100250c[i]=5 and e100250cb[i]>0 and e98429[i]>0 then
          othpay1[i]=(e100250[i]/e100250cb[i])/(e98429[i]*4.3);
    /* missing value */
      if e100250c[i] in (3,4,5) and -4<e98429[i]<=0 then othpay1[i]=-3;
```

```
  end;
   if e100234[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98429[i]=0 then othpay1[i]=-3;
   if e100234[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98429[i]<0 then
            othpay1[i]=e98429[i];

  /* non-hourly wage at the beginning, without overtime*/
  if e100234[i]=2 and e100250[i]>=0 and e100250e[i]>0 and e98402[i]>0 then
            othpay1[i]=e100250[i]*e100250e[i]/e98402[i];
  if e100234[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e100250[i]>=0 and e98402[i]>0
            then othpay1[i]=e100250[i]/e98402[i];
  if e100234[i]=4 and e100250[i]>=0 and e98402[i]>0 then othpay1[i]=e100250[i]/(2*e98402[i]);
  if e100234[i]=5 and e100250[i]>=0 and e98402[i]>0 then othpay1[i]=e100250[i]/(4.3*e98402[i]);
  if e100234[i]=6 and e100250[i]>=0 and e98402[i]>0 and e99500[i]>0 then
            othpay1[i]=e100250[i]/(e99500[i]*e98402[i]);
  if e100234[i]=8 and e100250[i]>=0 and e98402[i]>0 then othpay1[i]=e100250[i]/(2.15*e98402[i]);
     if e100234[i] in (12,13) and e100250[i]>=0 then do;
      /* condition based on the time unit to finish per othpay1. */
       if e100250c[i]=3 and e100250cb[i]>0 and e100250e[i]>=0 and e98402[i]>0 then
         othpay1[i]=(e100250[i]/e100250cb[i])*e100250e[i]/e98402[i];
       if e100250c[i]=4 and e100250cb[i]>0 and e98402[i]>0 then othpay1[i]=(e100250[i]/e100250cb[i])/e98402[i];
       if e100250c[i]=5 and e100250cb[i]>0 and e98402[i]>0 then
            othpay1[i]=(e100250[i]/e100250cb[i])/(e98402[i]*4.3);
     /* missing value */
       if e100250c[i] in (3,4,5) and -4<e98402[i]<=0 then othpay1[i]=-3;
  end;
   if e100234[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98402[i]=0 then othpay1[i]=-3;
   if e100234[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98402[i]<0 then
            othpay1[i]=e98402[i];

  if e100234[i]=6 and e99500[i] le 0 then othpay1[i]=-3;
  if e100234[i]=6 and -4<e99500[i]<0 then othpay1[i]=e99500[i];

 end;
 end;

/********* for commissions *********/

  /*report hourly wage at the beginning*/
   do i=1 to 9;
  if e102052[i]=1 and e102142[i]>=0 then othpay2[i]=e102142[i];
  if e88000[i]>0 and e102052[i]=2 and e10214b2[i]>0 and e102142[i]>=0 then
            othpay2[i]=(e102142[i]*e10214b2[i])/e88000[i];
  if e88000[i]>0 and e102052[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e102142[i]>=0
            then othpay2[i]=e102142[i]/e88000[i];
  if e88000[i]>0 and e102052[i]=4 and e102142[i]>=0 then othpay2[i]=e102142[i]/(2*e88000[i]);
  if e88000[i]>0 and e102052[i]=5 and e102142[i]>=0 then othpay2[i]=e102142[i]/(4.3*e88000[i]);
  if e88000[i]>0 and e102052[i]=6 and e102142[i]>=0 and e99500[i]>0 then
            othpay2[i]=e102142[i]/(e99500[i]*e88000[i]);
  if e88000[i]>0 and e102052[i]=8 and e102142[i]>=0 then othpay2[i]=e102142[i]/(2.15*e88000[i]);
  if e102052[i] in (9,14) then othpay2[i]=0;
  if e102052[i] in (12,13) and e102142[i]>=0 then do;
     /* condition based on the time unit to finish per othpay2. */
       if e10214aaa2[i]=1 and e10214aab2[i]>0 then othpay2[i]=(e102142[i]/e10214aab2[i])*60;
       if e10214aaa2[i]=2 and e10214aab2[i]>0 then othpay2[i]=e102142[i]/e10214aab2[i];
       if e10214aaa2[i]=3 and e10214aab2[i]>0 and e10214b2[i]>=0 and e88000[i]>0 then
         othpay2[i]=(e102142[i]/e10214aab2[i])*e10214b2[i]/e88000[i];
       if e10214aaa2[i]=4 and e10214aab2[i]>0 and e88000[i]>0 then othpay2[i]=(e102142[i]/e10214aab2[i])/e88000[i];
```

```
    if e10214aaa2[i]=5 and e10214aab2[i]>0 and e88000[i]>0 then
        othpay2[i]=(e102142[i]/e10214aab2[i])/(e88000[i]*4.3);
  /* missing value */
    if -4<e10214aaa2[i]<0 or -4<e10214aab2[i]<=0 then othpay2[i]=-3;
    if e10214aaa2[i] in (3,4,5) and -4<e88000[i]<=0 then othpay2[i]=-3;
    if e10214aaa2[i]=3 and e10214b2[i]<0 then othpay2[i]=-3;
end;
 if e102052[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e102142[i]<0 then
        othpay2[i]=e102142[i];
 if e102052[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e88000[i]=0 then othpay2[i]=-3;
 if e102052[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e88000[i]<0 then
        othpay2[i]=e88000[i];
 if e102052[i]=2 and e10214b2[i] le 0 then othpay2[i]=-3;
 if e102052[i]=2 and -4<e10214b2[i]<0 then othpay2[i]=e10214b2[i];

end;

/* non-hourly wage at the beginning, with overtime*/
do i=1 to 9;
if e98429[i]>0 and e102052[i]=2 and e10214b2[i]>0 and e102142[i]>=0 then
        othpay2[i]=(e102142[i]*e10214b2[i])/e98429[i];
if e98429[i]>0 and e102052[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e102142[i]>=0
        then othpay2[i]=e102142[i]/e98429[i];
if e98429[i]>0 and e102052[i]=4 and e102142[i]>=0 then othpay2[i]=e102142[i]/(2*e98429[i]);
if e98429[i]>0 and e102052[i]=5 and e102142[i]>=0 then othpay2[i]=e102142[i]/(4.3*e98429[i]);
if e98429[i]>0 and e102052[i]=6 and e102142[i]>=0 and e99500[i]>0 then
        othpay2[i]=e102142[i]/(e99500[i]*e98429[i]);
if e98429[i]>0 and e102052[i]=8 and e102142[i]>=0 then othpay2[i]=e102142[i]/(2.15*e98429[i]);
if e102052[i] in (12,13) and e102142[i]>=0 then do;
   /* condition based on the time unit to finish per othpay2. */
    if e10214aaa2[i]=3 and e10214aab2[i]>0 and e10214b2[i]>=0 and e98429[i]>0 then
      othpay2[i]=(e102142[i]/e10214aab2[i])*e10214b2[i]/e98429[i];
    if e10214aaa2[i]=4 and e10214aab2[i]>0 and e98429[i]>0 then othpay2[i]=(e102142[i]/e10214aab2[i])/e98429[i];
    if e10214aaa2[i]=5 and e10214aab2[i]>0 and e98429[i]>0 then
        othpay2[i]=(e102142[i]/e10214aab2[i])/(e98429[i]*4.3);
  /* missing value */
    if e10214aaa2[i] in (3,4,5) and -4<e98429[i]<=0 then othpay2[i]=-3;
end;
 if e102052[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98429[i]=0 then othpay2[i]=-3;
 if e102052[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98429[i]<0 then
        othpay2[i]=e98429[i];

end;

/* non-hourly wage at the beginning, without overtime*/
do i=1 to 9;
if e98402[i]>0 and e102052[i]=2 and e10214b2[i]>0 and e102142[i]>=0 then
        othpay2[i]=(e102142[i]*e10214b2[i])/e98402[i];
if e98402[i]>0 and e102052[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e102142[i]>=0
        then othpay2[i]=e102142[i]/e98402[i];
if e98402[i]>0 and e102052[i]=4 and e102142[i]>=0 then othpay2[i]=e102142[i]/(2*e98402[i]);
if e98402[i]>0 and e102052[i]=5 and e102142[i]>=0 then othpay2[i]=e102142[i]/(4.3*e98402[i]);
if e98402[i]>0 and e102052[i]=6 and e102142[i]>=0 and e99500[i]>0 then
        othpay2[i]=e102142[i]/(e99500[i]*e98402[i]);
if e98402[i]>0 and e102052[i]=8 and e102142[i]>=0 then othpay2[i]=e102142[i]/(2.15*e98402[i]);
if e102052[i] in (7,9,12,13,15,16,17,14,18,19,21,22,23,24,25,26,28) then othpf2[i]=othpf2[i]+1;
 if e102052[i] in (12,13) and e102142[i]>=0 then do;
```

```
      /* condition based on the time unit to finish per othpay2. */
       if e10214aaa2[i]=3 and e10214aab2[i]>0 and e10214b2[i]>=0 and e98402[i]>0 then
         othpay2[i]=(e102142[i]/e10214aab2[i])*e10214b2[i]/e98402[i];
       if e10214aaa2[i]=4 and e10214aab2[i]>0 and e98402[i]>0 then othpay2[i]=(e102142[i]/e10214aab2[i])/e98402[i];
       if e10214aaa2[i]=5 and e10214aab2[i]>0 and e98402[i]>0 then
            othpay2[i]=(e102142[i]/e10214aab2[i])/(e98402[i]*4.3);
     /* missing value */
       if e10214aaa2[i] in (3,4,5) and -4<e98402[i]<=0 then othpay2[i]=-3;
   end;
    if e102052[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98402[i]=0 then othpay2[i]=-3;
    if e102052[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98402[i]<0 then
            othpay2[i]=e98402[i];
   if e102052[i]=6 and e99500[i] le 0 then othpay2[i]=-3;
   if e102052[i]=6 and -4<e99500[i]<0 then othpay2[i]=e99500[i];

   end;

/* if commissions is corrected in the later part */
   do i=1 to 9;
   if e102256[i]=1 then do;
   /*report hourly wage at the beginning*/
   if e100235[i]=1 and e100251[i]>=0 then othpay2[i]=e100251[i];
   if e100235[i]=2 and e100251[i]>=0 and e100251e[i]>0 and e88000[i]>0 then
            othpay2[i]=e100251[i]*e100251e[i]/e88000[i];
   if e100235[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e100251[i]>=0 and e88000[i]>0
            then othpay2[i]=e100251[i]/e88000[i];
   if e100235[i]=4 and e100251[i]>=0 and e88000[i]>0 then othpay2[i]=e100251[i]/(2*e88000[i]);
   if e100235[i]=5 and e100251[i]>=0 and e88000[i]>0 then othpay2[i]=e100251[i]/(4.3*e88000[i]);
   if e100235[i]=6 and e100251[i]>=0 and e88000[i]>0 and e99500[i]>0 then
            othpay2[i]=e100251[i]/(e99500[i]*e88000[i]);
   if e100235[i]=8 and e100251[i]>=0 and e88000[i]>0 then othpay2[i]=e100251[i]/(2.15*e88000[i]);
   if e100235[i] in (9,14) then othpay2[i]=0;
   if e100235[i] in (12,13) and e100251[i]>=0 then do;
      /* condition based on the time unit to finish per othpay2. */
       if e100251c[i]=1 and e100251cb[i]>0 then othpay2[i]=(e100251[i]/e100251cb[i])*60;
       if e100251c[i]=2 and e100251cb[i]>0 then othpay2[i]=e100251[i]/e100251cb[i];
       if e100251c[i]=3 and e100251cb[i]>0 and e100251e[i]>=0 and e88000[i]>0 then
         othpay2[i]=(e100251[i]/e100251cb[i])*e100251e[i]/e88000[i];
       if e100251c[i]=4 and e100251cb[i]>0 and e88000[i]>0 then othpay2[i]=(e100251[i]/e100251cb[i])/e88000[i];
       if e100251c[i]=5 and e100251cb[i]>0 and e88000[i]>0 then
            othpay2[i]=(e100251[i]/e100251cb[i])/(e88000[i]*4.3);
     /* missing value */
       if -4<e100251c[i]<0 or -4<e100251cb[i]<=0 then othpay2[i]=-3;
       if e100251c[i] in (3,4,5) and -4<e88000[i]<=0 then othpay2[i]=-3;
       if e100251c[i]=3 and e100251e[i]<0 then othpay2[i]=-3;
   end;
    if e100235[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e100251[i]<0 then
            othpay2[i]=e100251[i];
    if e100235[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e88000[i]=0 then othpay2[i]=-3;
    if e100235[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e88000[i]<0 then
            othpay2[i]=e88000[i];
    if e100235[i]=2 and e100251e[i] le 0 then othpay2[i]=-3;
    if e100235[i]=2 and -4<e100251e[i]<0 then othpay2[i]=e100251e[i];

   /* non-hourly wage at the beginning, with overtime*/
   if e100235[i]=2 and e100251[i]>=0 and e100251e[i]>0 and e98429[i]>0 then
            othpay2[i]=e100251[i]*e100251e[i]/e98429[i];
```

```
  if e100235[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e100251[i]>=0 and e98429[i]>0
          then othpay2[i]=e100251[i]/e98429[i];
  if e100235[i]=4 and e100251[i]>=0 and e98429[i]>0 then othpay2[i]=e100251[i]/(2*e98429[i]);
  if e100235[i]=5 and e100251[i]>=0 and e98429[i]>0 then othpay2[i]=e100251[i]/(4.3*e98429[i]);
  if e100235[i]=6 and e100251[i]>=0 and e98429[i]>0 and e99500[i]>0 then
          othpay2[i]=e100251[i]/(e99500[i]*e98429[i]);
  if e100235[i]=8 and e100251[i]>=0 and e98429[i]>0 then othpay2[i]=e100251[i]/(2.15*e98429[i]);
   if e100235[i] in (12,13) and e100251[i]>=0 then do;
     /* condition based on the time unit to finish per othpay2. */
      if e100251c[i]=3 and e100251cb[i]>0 and e100251e[i]>=0 and e98429[i]>0 then
        othpay2[i]=(e100251[i]/e100251cb[i])*e100251e[i]/e98429[i];
      if e100251c[i]=4 and e100251cb[i]>0 and e98429[i]>0 then othpay2[i]=(e100251[i]/e100251cb[i])/e98429[i];
      if e100251c[i]=5 and e100251cb[i]>0 and e98429[i]>0 then
          othpay2[i]=(e100251[i]/e100251cb[i])/(e98429[i]*4.3);
    /* missing value */
      if e100251c[i] in (3,4,5) and -4<e98429[i]<=0 then othpay2[i]=-3;
  end;
   if e100235[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98429[i]=0 then othpay2[i]=-3;
   if e100235[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98429[i]<0 then
          othpay2[i]=e98429[i];

  /* non-hourly wage at the beginning, without overtime*/
  if e100235[i]=2 and e100251[i]>=0 and e100251e[i]>0 and e98402[i]>0 then
          othpay2[i]=e100251[i]*e100251e[i]/e98402[i];
  if e100235[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e100251[i]>=0 and e98402[i]>0
          then othpay2[i]=e100251[i]/e98402[i];
  if e100235[i]=4 and e100251[i]>=0 and e98402[i]>0 then othpay2[i]=e100251[i]/(2*e98402[i]);
  if e100235[i]=5 and e100251[i]>=0 and e98402[i]>0 then othpay2[i]=e100251[i]/(4.3*e98402[i]);
  if e100235[i]=6 and e100251[i]>=0 and e98402[i]>0 and e99500[i]>0 then
          othpay2[i]=e100251[i]/(e99500[i]*e98402[i]);
  if e100235[i]=8 and e100251[i]>=0 and e98402[i]>0 then othpay2[i]=e100251[i]/(2.15*e98402[i]);
  if e100235[i] in (12,13) and e100251[i]>=0 then do;
     /* condition based on the time unit to finish per othpay2. */
      if e100251c[i]=3 and e100251cb[i]>0 and e100251e[i]>=0 and e98402[i]>0 then
        othpay2[i]=(e100251[i]/e100251cb[i])*e100251e[i]/e98402[i];
      if e100251c[i]=4 and e100251cb[i]>0 and e98402[i]>0 then othpay2[i]=(e100251[i]/e100251cb[i])/e98402[i];
      if e100251c[i]=5 and e100251cb[i]>0 and e98402[i]>0 then
          othpay2[i]=(e100251[i]/e100251cb[i])/(e98402[i]*4.3);
    /* missing value */
      if e100251c[i] in (3,4,5) and -4<e98402[i]<=0 then othpay2[i]=-3;
  end;
   if e100235[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98402[i]=0 then othpay2[i]=-3;
   if e100235[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98402[i]<0 then
          othpay2[i]=e98402[i];
  if e100235[i]=6 and e99500[i] le 0 then othpay2[i]=-3;
  if e100235[i]=6 and -4<e99500[i]<0 then othpay2[i]=e99500[i];
 end;
 end;

/******** for bonuses ********/

  /*report hourly wage at the beginning*/
  do i=1 to 9;
  if e102053[i]=1 and e102143[i]>=0 then othpay3[i]=e102143[i];
  if e88000[i]>0 and e102053[i]=2 and e10214b3[i]>0 and e102143[i]>=0 then
          othpay3[i]=(e102143[i]*e10214b3[i])/e88000[i];
```

```
if e88000[i]>0 and e102053[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e102143[i]>=0
        then othpay3[i]=e102143[i]/e88000[i];
if e88000[i]>0 and e102053[i]=4 and e102143[i]>=0 then othpay3[i]=e102143[i]/(2*e88000[i]);
if e88000[i]>0 and e102053[i]=5 and e102143[i]>=0 then othpay3[i]=e102143[i]/(4.3*e88000[i]);
if e88000[i]>0 and e102053[i]=6 and e102143[i]>=0 and e99500[i]>0 then
        othpay3[i]=e102143[i]/(e99500[i]*e88000[i]);
if e88000[i]>0 and e102053[i]=8 and e102143[i]>=0 then othpay3[i]=e102143[i]/(2.15*e88000[i]);
if e102053[i] in (9,14) then othpay2[i]=0;
if e102053[i] in (12,13) and e102143[i]>=0 then do;
   /* condition based on the time unit to finish per othpay2. */
    if e10214aaa3[i]=1 and e10214aab3[i]>0 then othpay3[i]=(e102143[i]/e10214aab3[i])*60;
    if e10214aaa3[i]=2 and e10214aab3[i]>0 then othpay3[i]=e102143[i]/e10214aab3[i];
    if e10214aaa3[i]=3 and e10214aab3[i]>0 and e10214b3[i]>=0 and e88000[i]>0 then
      othpay3[i]=(e102143[i]/e10214aab3[i])*e10214b3[i]/e88000[i];
    if e10214aaa3[i]=4 and e10214aab3[i]>0 and e88000[i]>0 then othpay3[i]=(e102143[i]/e10214aab3[i])/e88000[i];
    if e10214aaa3[i]=5 and e10214aab3[i]>0 and e88000[i]>0 then
        othpay3[i]=(e102143[i]/e10214aab3[i])/(e88000[i]*4.3);
  /* missing value */
    if -4<e10214aaa3[i]<0 or -4<e10214aab3[i]<=0 then othpay3[i]=-3;
    if e10214aaa3[i] in (3,4,5) and -4<e88000[i]<=0 then othpay3[i]=-3;
    if e10214aaa3[i]=3 and e10214b3[i]<0 then othpay3[i]=-3;
end;
 if e102053[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e102143[i]<0 then
        othpay3[i]=e102143[i];
 if e102053[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e88000[i]=0 then othpay3[i]=-3;
 if e102053[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e88000[i]<0 then
        othpay3[i]=e88000[i];
 if e102053[i]=2 and e10214b3[i] le 0 then othpay3[i]=-3;
 if e102053[i]=2 and -4<e10214b3[i]<0 then othpay3[i]=e10214b3[i];

end;

/* non-hourly wage at the beginning, with overtime*/
do i=1 to 9;
if e98429[i]>0 and e102053[i]=2 and e10214b3[i]>0 and e102143[i]>=0 then
        othpay3[i]=(e102143[i]*e10214b3[i])/e98429[i];
if e98429[i]>0 and e102053[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e102143[i]>=0
        then othpay3[i]=e102143[i]/e98429[i];
if e98429[i]>0 and e102053[i]=4 and e102143[i]>=0 then othpay3[i]=e102143[i]/(2*e98429[i]);
if e98429[i]>0 and e102053[i]=5 and e102143[i]>=0 then othpay3[i]=e102143[i]/(4.3*e98429[i]);
if e98429[i]>0 and e102053[i]=6 and e102143[i]>=0 and e99500[i]>0 then
        othpay3[i]=e102143[i]/(e99500[i]*e98429[i]);
if e98429[i]>0 and e102053[i]=8 and e102143[i]>=0 then othpay3[i]=e102143[i]/(2.15*e98429[i]);
if e102053[i] in (12,13) and e102143[i]>=0 then do;
   /* condition based on the time unit to finish per othpay3. */
    if e10214aaa3[i]=3 and e10214aab3[i]>0 and e10214b3[i]>=0 and e98429[i]>0 then
      othpay3[i]=(e102143[i]/e10214aab3[i])*e10214b3[i]/e98429[i];
    if e10214aaa3[i]=4 and e10214aab3[i]>0 and e98429[i]>0 then othpay3[i]=(e102143[i]/e10214aab3[i])/e98429[i];
    if e10214aaa3[i]=5 and e10214aab3[i]>0 and e98429[i]>0 then
        othpay3[i]=(e102143[i]/e10214aab3[i])/(e98429[i]*4.3);
  /* missing value */
    if e10214aaa3[i] in (3,4,5) and -4<e98429[i]<=0 then othpay3[i]=-3;
end;
 if e102053[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98429[i]=0 then othpay3[i]=-3;
 if e102053[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98429[i]<0 then
        othpay3[i]=e98429[i];
end;
```

```
   /* non-hourly wage at the beginning, without overtime*/
   do i=1 to 9;
   if e98402[i]>0 and e102053[i]=2 and e10214b3[i]>0 and e102143[i]>=0 then
            othpay3[i]=(e102143[i]*e10214b3[i])/e98402[i];
   if e98402[i]>0 and e102053[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e102143[i]>=0
            then othpay3[i]=e102143[i]/e98402[i];
   if e98402[i]>0 and e102053[i]=4 and e102143[i]>=0 then othpay3[i]=e102143[i]/(2*e98402[i]);
   if e98402[i]>0 and e102053[i]=5 and e102143[i]>=0 then othpay3[i]=e102143[i]/(4.3*e98402[i]);
   if e98402[i]>0 and e102053[i]=6 and e102143[i]>=0 and e99500[i]>0 then
            othpay3[i]=e102143[i]/(e99500[i]*e98402[i]);
   if e98402[i]>0 and e102053[i]=8 and e102143[i]>=0 then othpay3[i]=e102143[i]/(2.15*e98402[i]);
   if e102053[i] in (7,9,12,13,15,16,17,14,18,19,21,22,23,24,25,26,28) then othpf3[i]=othpf3[i]+1;
   if e102053[i] in (12,13) and e102143[i]>=0 then do;
      /* condition based on the time unit to finish per othpay3. */
       if e10214aaa3[i]=3 and e10214aab3[i]>0 and e10214b3[i]>=0 and e98402[i]>0 then
         othpay3[i]=(e102143[i]/e10214aab3[i])*e10214b3[i]/e98402[i];
       if e10214aaa3[i]=4 and e10214aab3[i]>0 and e98402[i]>0 then othpay3[i]=(e102143[i]/e10214aab3[i])/e98402[i];
       if e10214aaa3[i]=5 and e10214aab3[i]>0 and e98402[i]>0 then
            othpay3[i]=(e102143[i]/e10214aab3[i])/(e98402[i]*4.3);
     /* missing value */
       if e10214aaa3[i] in (3,4,5) and -4<e98402[i]<=0 then othpay3[i]=-3;
   end;
    if e102053[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98402[i]=0 then othpay3[i]=-3;
    if e102053[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98402[i]<0 then
            othpay3[i]=e98402[i];
   if e102053[i]=6 and e99500[i] le 0 then othpay3[i]=-3;
   if e102053[i]=6 and -4<e99500[i]<0 then othpay3[i]=e99500[i];

   end;

/* if bonus is corrected in the later part */
   do i=1 to 9;
   if e102257[i]=1 then do;
   /*report hourly wage at the beginning*/
   if e100236[i]=1 and e100252[i]>=0 then othpay3[i]=e100252[i];
   if e100236[i]=2 and e100252[i]>=0 and e100252e[i]>0 and e88000[i]>0 then
            othpay3[i]=e100252[i]*e100252e[i]/e88000[i];
   if e100236[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e100252[i]>=0 and e88000[i]>0
            then othpay3[i]=e100252[i]/e88000[i];
   if e100236[i]=4 and e100252[i]>=0 and e88000[i]>0 then othpay3[i]=e100252[i]/(2*e88000[i]);
   if e100236[i]=5 and e100252[i]>=0 and e88000[i]>0 then othpay3[i]=e100252[i]/(4.3*e88000[i]);
   if e100236[i]=6 and e100252[i]>=0 and e88000[i]>0 and e99500[i]>0 then
            othpay3[i]=e100252[i]/(e99500[i]*e88000[i]);
   if e100236[i]=8 and e100252[i]>=0 and e88000[i]>0 then othpay3[i]=e100252[i]/(2.15*e88000[i]);
   if e100236[i] in (9,14) then othpay3[i]=0;
   if e100236[i] in (12,13) and e100252[i]>=0 then do;
      /* condition based on the time unit to finish per othpay3. */
       if e100252c[i]=1 and e100252cb[i]>0 then othpay3[i]=(e100252[i]/e100252cb[i])*60;
       if e100252c[i]=2 and e100252cb[i]>0 then othpay3[i]=e100252[i]/e100252cb[i];
       if e100252c[i]=3 and e100252cb[i]>0 and e100252e[i]>=0 and e88000[i]>0 then
         othpay3[i]=(e100252[i]/e100252cb[i])*e100252e[i]/e88000[i];
       if e100252c[i]=4 and e100252cb[i]>0 and e88000[i]>0 then othpay3[i]=(e100252[i]/e100252cb[i])/e88000[i];
       if e100252c[i]=5 and e100252cb[i]>0 and e88000[i]>0 then
            othpay3[i]=(e100252[i]/e100252cb[i])/(e88000[i]*4.3);
     /* missing value */
       if -4<e100252c[i]<0 or -4<e100252cb[i]<=0 then othpay3[i]=-3;
```

```
    if e100252c[i] in (3,4,5) and -4<e88000[i]<=0 then othpay3[i]=-3;
    if e100252c[i]=3 and e100252e[i]<0 then othpay3[i]=-3;
end;
 if e100236[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e100252[i]<0 then
        othpay3[i]=e100252[i];
 if e100236[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e88000[i]=0 then othpay3[i]=-3;
 if e100236[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e88000[i]<0 then
        othpay3[i]=e88000[i];
 if e100236[i]=2 and e100252e[i] le 0 then othpay3[i]=-3;
 if e100236[i]=2 and -4<e100252e[i]<0 then othpay3[i]=e100252e[i];

/* non-hourly wage at the beginning, with overtime*/
if e100236[i]=2 and e100252[i]>=0 and e100252e[i]>0 and e98429[i]>0 then
        othpay3[i]=e100252[i]*e100252e[i]/e98429[i];
if e100236[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e100252[i]>=0 and e98429[i]>0
        then othpay3[i]=e100252[i]/e98429[i];
if e100236[i]=4 and e100252[i]>=0 and e98429[i]>0 then othpay3[i]=e100252[i]/(2*e98429[i]);
if e100236[i]=5 and e100252[i]>=0 and e98429[i]>0 then othpay3[i]=e100252[i]/(4.3*e98429[i]);
if e100236[i]=6 and e100252[i]>=0 and e98429[i]>0 and e99500[i]>0 then
        othpay3[i]=e100252[i]/(e99500[i]*e98429[i]);
if e100236[i]=8 and e100252[i]>=0 and e98429[i]>0 then othpay3[i]=e100252[i]/(2.15*e98429[i]);
if e100236[i] in (12,13) and e100252[i]>=0 then do;
   /* condition based on the time unit to finish per othpay3. */
    if e100252c[i]=3 and e100252cb[i]>0 and e100252e[i]>=0 and e98429[i]>0 then
      othpay3[i]=(e100252[i]/e100252cb[i])*e100252e[i]/e98429[i];
    if e100252c[i]=4 and e100252cb[i]>0 and e98429[i]>0 then othpay3[i]=(e100252[i]/e100252cb[i])/e98429[i];
    if e100252c[i]=5 and e100252cb[i]>0 and e98429[i]>0 then
        othpay3[i]=(e100252[i]/e100252cb[i])/(e98429[i]*4.3);
  /* missing value */
    if e100252c[i] in (3,4,5) and -4<e98429[i]<=0 then othpay3[i]=-3;
end;
 if e100236[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98429[i]=0 then othpay3[i]=-3;
 if e100236[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98429[i]<0 then
        othpay3[i]=e98429[i];

/* non-hourly wage at the beginning, without overtime*/
if e100236[i]=2 and e100252[i]>=0 and e100252e[i]>0 and e98402[i]>0 then
        othpay3[i]=e100252[i]*e100252e[i]/e98402[i];
if e100236[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e100252[i]>=0 and e98402[i]>0
        then othpay3[i]=e100252[i]/e98402[i];
if e100236[i]=4 and e100252[i]>=0 and e98402[i]>0 then othpay3[i]=e100252[i]/(2*e98402[i]);
if e100236[i]=5 and e100252[i]>=0 and e98402[i]>0 then othpay3[i]=e100252[i]/(4.3*e98402[i]);
if e100236[i]=6 and e100252[i]>=0 and e98402[i]>0 and e99500[i]>0 then
        othpay3[i]=e100252[i]/(e99500[i]*e98402[i]);
if e100236[i]=8 and e100252[i]>=0 and e98402[i]>0 then othpay3[i]=e100252[i]/(2.15*e98402[i]);
if e100236[i] in (12,13) and e100252[i]>=0 then do;
   /* condition based on the time unit to finish per othpay3. */
    if e100252c[i]=3 and e100252cb[i]>0 and e100252e[i]>=0 and e98402[i]>0 then
      othpay3[i]=(e100252[i]/e100252cb[i])*e100252e[i]/e98402[i];
    if e100252c[i]=4 and e100252cb[i]>0 and e98402[i]>0 then othpay3[i]=(e100252[i]/e100252cb[i])/e98402[i];
    if e100252c[i]=5 and e100252cb[i]>0 and e98402[i]>0 then
        othpay3[i]=(e100252[i]/e100252cb[i])/(e98402[i]*4.3);
  /* missing value */
    if e100252c[i] in (3,4,5) and -4<e98402[i]<=0 then othpay3[i]=-3;
end;
 if e100236[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98402[i]=0 then othpay3[i]=-3;
```

```
 if e100236[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98402[i]<0 then
        othpay3[i]=e98402[i];
 if e100236[i]=6 and e99500[i] le 0 then othpay3[i]=-3;
 if e100236[i]=6 and -4<e99500[i]<0 then othpay3[i]=e99500[i];
 end;
 end;

/********* for incentive pay *********/

 /*report hourly wage at the beginning*/
 do i=1 to 9;
 if e102054[i]=1 and e102144[i]>=0 then othpay4[i]=e102144[i];
 if e88000[i]>0 and e102054[i]=2 and e10214b4[i]>0 and e102144[i]>=0 then
        othpay4[i]=(e102144[i]*e10214b4[i])/e88000[i];
 if e88000[i]>0 and e102054[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e102144[i]>=0
        then othpay4[i]=e102144[i]/e88000[i];
 if e88000[i]>0 and e102054[i]=4 and e102144[i]>=0 then othpay4[i]=e102144[i]/(2*e88000[i]);
 if e88000[i]>0 and e102054[i]=5 and e102144[i]>=0 then othpay4[i]=e102144[i]/(4.3*e88000[i]);
 if e88000[i]>0 and e102054[i]=6 and e102144[i]>=0 and e99500[i]>0 then
        othpay4[i]=e102144[i]/(e99500[i]*e88000[i]);
 if e88000[i]>0 and e102054[i]=8 and e102144[i]>=0 then othpay4[i]=e102144[i]/(2.15*e88000[i]);
 if e102054[i] in (9, 14) then othpay4[i]=0;
 if e102054[i] in (12,13) and e102144[i]>=0 then do;
    /* condition based on the time unit to finish per othpay2. */
     if e10214aaa4[i]=1 and e10214aab4[i]>0 then othpay4[i]=(e102144[i]/e10214aab4[i])*60;
     if e10214aaa4[i]=2 and e10214aab4[i]>0 then othpay4[i]=e102144[i]/e10214aab4[i];
     if e10214aaa4[i]=3 and e10214aab4[i]>0 and e10214b4[i]>=0 and e88000[i]>0 then
       othpay4[i]=(e102144[i]/e10214aab4[i])*e10214b4[i]/e88000[i];
     if e10214aaa4[i]=4 and e10214aab4[i]>0 and e88000[i]>0 then othpay4[i]=(e102144[i]/e10214aab4[i])/e88000[i];
     if e10214aaa4[i]=5 and e10214aab4[i]>0 and e88000[i]>0 then
        othpay4[i]=(e102144[i]/e10214aab4[i])/(e88000[i]*4.3);
   /* missing value */
     if -4<e10214aaa4[i]<0 or -4<e10214aab4[i]<=0 then othpay4[i]=-3;
     if e10214aaa4[i] in (3,4,5) and -4<e88000[i]<=0 then othpay4[i]=-3;
     if e10214aaa4[i]=3 and e10214b4[i]<0 then othpay4[i]=-3;
 end;
 if e102054[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e102144[i]<0 then
        othpay4[i]=e102144[i];
 if e102054[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e88000[i]=0 then othpay4[i]=-3;
 if e102054[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e88000[i]<0 then
        othpay4[i]=e88000[i];
 if e102054[i]=2 and e10214b4[i] le 0 then othpay4[i]=-3;
 if e102054[i]=2 and -4<e10214b4[i]<0 then othpay4[i]=e10214b4[i];

 end;

 /* non-hourly wage at the beginning, with overtime*/
 do i=1 to 9;
 if e98429[i]>0 and e102054[i]=2 and e10214b4[i]>0 and e102144[i]>=0 then
        othpay4[i]=(e102144[i]*e10214b4[i])/e98429[i];
 if e98429[i]>0 and e102054[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e102144[i]>=0
        then othpay4[i]=e102144[i]/e98429[i];
 if e98429[i]>0 and e102054[i]=4 and e102144[i]>=0 then othpay4[i]=e102144[i]/(2*e98429[i]);
 if e98429[i]>0 and e102054[i]=5 and e102144[i]>=0 then othpay4[i]=e102144[i]/(4.3*e98429[i]);
 if e98429[i]>0 and e102054[i]=6 and e102144[i]>=0 and e99500[i]>0 then
        othpay4[i]=e102144[i]/(e99500[i]*e98429[i]);
 if e98429[i]>0 and e102054[i]=8 and e102144[i]>=0 then othpay4[i]=e102144[i]/(2.15*e98429[i]);
```

```
 if e102054[i] in (12,13) and e102144[i]>=0 then do;
    /* condition based on the time unit to finish per othpay2. */
     if e10214aaa4[i]=3 and e10214aab4[i]>0 and e10214b4[i]>=0 and e98429[i]>0 then
       othpay4[i]=(e102144[i]/e10214aab4[i])*e10214b4[i]/e98429[i];
     if e10214aaa4[i]=4 and e10214aab4[i]>0 and e98429[i]>0 then othpay4[i]=(e102144[i]/e10214aab4[i])/e98429[i];
     if e10214aaa4[i]=5 and e10214aab4[i]>0 and e98429[i]>0 then
          othpay4[i]=(e102144[i]/e10214aab4[i])/(e98429[i]*4.3);
   /* missing value */
     if e10214aaa4[i] in (3,4,5) and -4<e98429[i]<=0 then othpay4[i]=-3;
 end;
  if e102054[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98429[i]=0 then othpay4[i]=-3;
  if e102054[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98429[i]<0 then
          othpay4[i]=e98429[i];
 end;

 /* non-hourly wage at the beginning, without overtime*/
 do i=1 to 9;
 if e98402[i]>0 and e102054[i]=2 and e10214b4[i]>0 and e102144[i]>=0 then
          othpay4[i]=(e102144[i]*e10214b4[i])/e98402[i];
 if e98402[i]>0 and e102054[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e102144[i]>=0
          then othpay4[i]=e102144[i]/e98402[i];
 if e98402[i]>0 and e102054[i]=4 and e102144[i]>=0 then othpay4[i]=e102144[i]/(2*e98402[i]);
 if e98402[i]>0 and e102054[i]=5 and e102144[i]>=0 then othpay4[i]=e102144[i]/(4.3*e98402[i]);
 if e98402[i]>0 and e102054[i]=6 and e102144[i]>=0 and e99500[i]>0 then
          othpay4[i]=e102144[i]/(e99500[i]*e98402[i]);
 if e98402[i]>0 and e102054[i]=8 and e102144[i]>=0 then othpay4[i]=e102144[i]/(2.15*e98402[i]);
 if e102054[i] in (7,9,12,13,15,16,17,14,18,19,21,22,23,24,25,26,28) then othpf4[i]=othpf4[i]+1;
   if e102054[i] in (12,13) and e102144[i]>=0 then do;
    /* condition based on the time unit to finish per othpay2. */
     if e10214aaa4[i]=3 and e10214aab4[i]>0 and e10214b4[i]>=0 and e98402[i]>0 then
       othpay4[i]=(e102144[i]/e10214aab4[i])*e10214b4[i]/e98402[i];
     if e10214aaa4[i]=4 and e10214aab4[i]>0 and e98402[i]>0 then othpay4[i]=(e102144[i]/e10214aab4[i])/e98402[i];
     if e10214aaa4[i]=5 and e10214aab4[i]>0 and e98402[i]>0 then
          othpay4[i]=(e102144[i]/e10214aab4[i])/(e98402[i]*4.3);
   /* missing value */
     if e10214aaa4[i] in (3,4,5) and -4<e98402[i]<=0 then othpay4[i]=-3;
 end;
  if e102054[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98402[i]=0 then othpay4[i]=-3;
  if e102054[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98402[i]<0 then
          othpay4[i]=e98402[i];
 if e102054[i]=6 and e99500[i] le 0 then othpay4[i]=-3;
 if e102054[i]=6 and -4<e99500[i]<0 then othpay4[i]=e99500[i];

 end;

/* if incentive pay is corrected in the later part */
 do i=1 to 9;
 if e102258[i]=1 then do;
 /*report hourly wage at the beginning*/
 if e100237[i]=1 and e100253[i]>=0 then othpay4[i]=e100253[i];
 if e100237[i]=2 and e100253[i]>=0 and e100253e[i]>0 and e88000[i]>0 then
          othpay4[i]=e100253[i]*e100253e[i]/e88000[i];
 if e100237[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e100253[i]>=0 and e88000[i]>0
          then othpay4[i]=e100253[i]/e88000[i];
 if e100237[i]=4 and e100253[i]>=0 and e88000[i]>0 then othpay4[i]=e100253[i]/(2*e88000[i]);
 if e100237[i]=5 and e100253[i]>=0 and e88000[i]>0 then othpay4[i]=e100253[i]/(4.3*e88000[i]);
```

```
if e100237[i]=6 and e100253[i]>=0 and e88000[i]>0 and e99500[i]>0 then
        othpay4[i]=e100253[i]/(e99500[i]*e88000[i]);
if e100237[i]=8 and e100253[i]>=0 and e88000[i]>0 then othpay4[i]=e100253[i]/(2.15*e88000[i]);
if e100237[i] in (9,14) then othpay4[i]=0;
  if e100237[i] in (12,13) and e100253[i]>=0 then do;
    /* condition based on the time unit to finish per othpay4. */
     if e100253c[i]=1 and e100253cb[i]>0 then othpay4[i]=(e100253[i]/e100253cb[i])*60;
     if e100253c[i]=2 and e100253cb[i]>0 then othpay4[i]=e100253[i]/e100253cb[i];
     if e100253c[i]=3 and e100253cb[i]>0 and e100253e[i]>=0 and e88000[i]>0 then
       othpay4[i]=(e100253[i]/e100253cb[i])*e100253e[i]/e88000[i];
     if e100253c[i]=4 and e100253cb[i]>0 and e88000[i]>0 then othpay4[i]=(e100253[i]/e100253cb[i])/e88000[i];
     if e100253c[i]=5 and e100253cb[i]>0 and e88000[i]>0 then
        othpay4[i]=(e100253[i]/e100253cb[i])/(e88000[i]*4.3);
   /* missing value */
     if -4<e100253c[i]<0 or -4<e100253cb[i]<=0 then othpay4[i]=-3;
     if e100253c[i] in (3,4,5) and -4<e88000[i]<=0 then othpay4[i]=-3;
     if e100253c[i]=3 and e100253e[i]<0 then othpay4[i]=-3;
end;
 if e100237[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e100253[i]<0 then
        othpay4[i]=e100253[i];
 if e100237[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e88000[i]=0 then othpay4[i]=-3;
 if e100237[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e88000[i]<0 then
        othpay4[i]=e88000[i];
 if e100237[i]=2 and e100253e[i] le 0 then othpay4[i]=-3;
 if e100237[i]=2 and -4<e100253e[i]<0 then othpay4[i]=e100253e[i];

/* non-hourly wage at the beginning, with overtime*/
if e100237[i]=2 and e100253[i]>=0 and e100253e[i]>0 and e98429[i]>0 then
        othpay4[i]=e100253[i]*e100253e[i]/e98429[i];
if e100237[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e100253[i]>=0 and e98429[i]>0
        then othpay4[i]=e100253[i]/e98429[i];
if e100237[i]=4 and e100253[i]>=0 and e98429[i]>0 then othpay4[i]=e100253[i]/(2*e98429[i]);
if e100237[i]=5 and e100253[i]>=0 and e98429[i]>0 then othpay4[i]=e100253[i]/(4.3*e98429[i]);
if e100237[i]=6 and e100253[i]>=0 and e98429[i]>0 and e99500[i]>0 then
        othpay4[i]=e100253[i]/(e99500[i]*e98429[i]);
if e100237[i]=8 and e100253[i]>=0 and e98429[i]>0 then othpay4[i]=e100253[i]/(2.15*e98429[i]);
  if e100237[i] in (12,13) and e100253[i]>=0 then do;
    /* condition based on the time unit to finish per othpay4. */
     if e100253c[i]=3 and e100253cb[i]>0 and e100253e[i]>=0 and e98429[i]>0 then
       othpay4[i]=(e100253[i]/e100253cb[i])*e100253e[i]/e98429[i];
     if e100253c[i]=4 and e100253cb[i]>0 and e98429[i]>0 then othpay4[i]=(e100253[i]/e100253cb[i])/e98429[i];
     if e100253c[i]=5 and e100253cb[i]>0 and e98429[i]>0 then
        othpay4[i]=(e100253[i]/e100253cb[i])/(e98429[i]*4.3);
   /* missing value */
     if e100253c[i] in (3,4,5) and -4<e98429[i]<=0 then othpay4[i]=-3;
end;
 if e100237[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98429[i]=0 then othpay4[i]=-3;
 if e100237[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98429[i]<0 then
        othpay4[i]=e98429[i];

/* non-hourly wage at the beginning, without overtime*/
if e100237[i]=2 and e100253[i]>=0 and e100253e[i]>0 and e98402[i]>0 then
        othpay4[i]=e100253[i]*e100253e[i]/e98402[i];
if e100237[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e100253[i]>=0 and e98402[i]>0
        then othpay4[i]=e100253[i]/e98402[i];
if e100237[i]=4 and e100253[i]>=0 and e98402[i]>0 then othpay4[i]=e100253[i]/(2*e98402[i]);
if e100237[i]=5 and e100253[i]>=0 and e98402[i]>0 then othpay4[i]=e100253[i]/(4.3*e98402[i]);
```

```
  if e100237[i]=6 and e100253[i]>=0 and e98402[i]>0 and e99500[i]>0 then
          othpay4[i]=e100253[i]/(e99500[i]*e98402[i]);
  if e100237[i]=8 and e100253[i]>=0 and e98402[i]>0 then othpay4[i]=e100253[i]/(2.15*e98402[i]);
  if e100237[i] in (12,13) and e100253[i]>=0 then do;
     /* condition based on the time unit to finish per othpay4. */
      if e100253c[i]=3 and e100253cb[i]>0 and e100253e[i]>=0 and e98402[i]>0 then
        othpay4[i]=(e100253[i]/e100253cb[i])*e100253e[i]/e98402[i];
      if e100253c[i]=4 and e100253cb[i]>0 and e98402[i]>0 then othpay4[i]=(e100253[i]/e100253cb[i])/e98402[i];
      if e100253c[i]=5 and e100253cb[i]>0 and e98402[i]>0 then
          othpay4[i]=(e100253[i]/e100253cb[i])/(e98402[i]*4.3);
    /* missing value */
      if e100253c[i] in (3,4,5) and -4<e98402[i]<=0 then othpay4[i]=-3;
  end;
   if e100237[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98402[i]=0 then othpay4[i]=-3;
   if e100237[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98402[i]<0 then
          othpay4[i]=e98402[i];
  if e100237[i]=6 and e99500[i] le 0 then othpay4[i]=-3;
  if e100237[i]=6 and -4<e99500[i]<0 then othpay4[i]=e99500[i];
 end;
 end;

/******** for others ********/

  /*report hourly wage at the beginning*/
  do i=1 to 9;
  if e102055[i]=1 and e102145[i]>=0 then othpay5[i]=e102145[i];
  if e88000[i]>0 and e102055[i]=2 and e10214b5[i]>0 and e102145[i]>=0 then
          othpay5[i]=(e102145[i]*e10214b5[i])/e88000[i];
  if e88000[i]>0 and e102055[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e102145[i]>=0
          then othpay5[i]=e102145[i]/e88000[i];
  if e88000[i]>0 and e102055[i]=4 and e102145[i]>=0 then othpay5[i]=e102145[i]/(2*e88000[i]);
  if e88000[i]>0 and e102055[i]=5 and e102145[i]>=0 then othpay5[i]=e102145[i]/(4.3*e88000[i]);
  if e88000[i]>0 and e102055[i]=6 and e102145[i]>=0 and e99500[i]>0 then
          othpay5[i]=e102145[i]/(e99500[i]*e88000[i]);
  if e88000[i]>0 and e102055[i]=8 and e102145[i]>=0 then othpay5[i]=e102145[i]/(2.15*e88000[i]);
  if e102055[i] in (9,14) then othpay5[i]=0;
  if e102055[i] in (12,13) and e102145[i]>=0 then do;
     /* condition based on the time unit to finish per othpay2. */
      if e10214aaa5[i]=1 and e10214aab5[i]>0 then othpay5[i]=(e102145[i]/e10214aab5[i])*60;
      if e10214aaa5[i]=2 and e10214aab5[i]>0 then othpay5[i]=e102145[i]/e10214aab5[i];
      if e10214aaa5[i]=3 and e10214aab5[i]>0 and e10214b5[i]>=0 and e88000[i]>0 then
        othpay5[i]=(e102145[i]/e10214aab5[i])*e10214b5[i]/e88000[i];
      if e10214aaa5[i]=4 and e10214aab5[i]>0 and e88000[i]>0 then othpay5[i]=(e102145[i]/e10214aab5[i])/e88000[i];
      if e10214aaa5[i]=5 and e10214aab5[i]>0 and e88000[i]>0 then
          othpay5[i]=(e102145[i]/e10214aab5[i])/(e88000[i]*4.3);
    /* missing value */
      if -4<e10214aaa5[i]<0 or -4<e10214aab5[i]<=0 then othpay5[i]=-3;
      if e10214aaa5[i] in (3,4,5) and -4<e88000[i]<=0 then othpay5[i]=-3;
      if e10214aaa5[i]=3 and e10214b5[i]<0 then othpay5[i]=-3;
  end;
   if e102055[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e102145[i]<0 then
          othpay5[i]=e102145[i];
   if e102055[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e88000[i]=0 then othpay5[i]=-3;
   if e102055[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e88000[i]<0 then
          othpay5[i]=e88000[i];
   if e102055[i]=2 and e10214b5[i] le 0 then othpay5[i]=-3;
   if e102055[i]=2 and -4<e10214b5[i]<0 then othpay5[i]=e10214b5[i];
```

```
end;

/* non-hourly wage at the beginning, with overtime*/
do i=1 to 9;
if e98429[i]>0 and e102055[i]=2 and e10214b5[i]>0 and e102145[i]>=0 then
        othpay5[i]=(e102145[i]*e10214b5[i])/e98429[i];
if e98429[i]>0 and e102055[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e102145[i]>=0
        then othpay5[i]=e102145[i]/e98429[i];
if e98429[i]>0 and e102055[i]=4 and e102145[i]>=0 then othpay5[i]=e102145[i]/(2*e98429[i]);
if e98429[i]>0 and e102055[i]=5 and e102145[i]>=0 then othpay5[i]=e102145[i]/(4.3*e98429[i]);
if e98429[i]>0 and e102055[i]=6 and e102145[i]>=0 and e99500[i]>0 then
        othpay5[i]=e102145[i]/(e99500[i]*e98429[i]);
if e98429[i]>0 and e102055[i]=8 and e102145[i]>=0 then othpay5[i]=e102145[i]/(2.15*e98429[i]);
if e102055[i] in (12,13) and e102145[i]>=0 then do;
   /* condition based on the time unit to finish per othpay2. */
    if e10214aaa5[i]=3 and e10214aab5[i]>0 and e10214b5[i]>=0 and e98429[i]>0 then
      othpay5[i]=(e102145[i]/e10214aab5[i])*e10214b5[i]/e98429[i];
    if e10214aaa5[i]=4 and e10214aab5[i]>0 and e98429[i]>0 then othpay5[i]=(e102145[i]/e10214aab5[i])/e98429[i];
    if e10214aaa5[i]=5 and e10214aab5[i]>0 and e98429[i]>0 then
        othpay5[i]=(e102145[i]/e10214aab5[i])/(e98429[i]*4.3);
  /* missing value */
    if e10214aaa5[i] in (3,4,5) and -4<e98429[i]<=0 then othpay5[i]=-3;
end;
 if e102055[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98429[i]=0 then othpay5[i]=-3;
 if e102055[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98429[i]<0 then
        othpay5[i]=e98429[i];

end;

/* non-hourly wage at the beginning, without overtime*/
do i=1 to 9;
if e98402[i]>0 and e102055[i]=1 and e102145[i]>=0 then othpay5[i]=e102145[i];
if e98402[i]>0 and e102055[i]=2 and e10214b5[i]>0 and e102145[i]>=0 then
        othpay5[i]=(e102145[i]*e10214b5[i])/e98402[i];
if e98402[i]>0 and e102055[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e102145[i]>=0
        then othpay5[i]=e102145[i]/e98402[i];
if e98402[i]>0 and e102055[i]=4 and e102145[i]>=0 then othpay5[i]=e102145[i]/(2*e98402[i]);
if e98402[i]>0 and e102055[i]=5 and e102145[i]>=0 then othpay5[i]=e102145[i]/(4.3*e98402[i]);
if e98402[i]>0 and e102055[i]=6 and e102145[i]>=0 and e99500[i]>0 then
        othpay5[i]=e102145[i]/(e99500[i]*e98402[i]);
if e98402[i]>0 and e102055[i]=8 and e102145[i]>=0 then othpay5[i]=e102145[i]/(2.15*e98402[i]);
if e102055[i] in (7,9,12,13,15,16,17,14,18,19,21,22,23,24,25,26,28) then othpf5[i]=othpf5[i]+1;
if e102055[i] in (9,14) then othpay5[i]=0;
 if e102055[i] in (12,13) and e102145[i]>=0 then do;
   /* condition based on the time unit to finish per othpay2. */
    if e10214aaa5[i]=3 and e10214aab5[i]>0 and e10214b5[i]>=0 and e98402[i]>0 then
      othpay5[i]=(e102145[i]/e10214aab5[i])*e10214b5[i]/e98402[i];
    if e10214aaa5[i]=4 and e10214aab5[i]>0 and e98402[i]>0 then othpay5[i]=(e102145[i]/e10214aab5[i])/e98402[i];
    if e10214aaa5[i]=5 and e10214aab5[i]>0 and e98402[i]>0 then
        othpay5[i]=(e102145[i]/e10214aab5[i])/(e98402[i]*4.3);
  /* missing value */
    if e10214aaa5[i] in (3,4,5) and -4<e98402[i]<=0 then othpay5[i]=-3;
end;
 if e102055[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98402[i]=0 then othpay5[i]=-3;
 if e102055[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98402[i]<0 then
        othpay5[i]=e98402[i];
if e102055[i]=6 and e99500[i] le 0 then othpay5[i]=-3;
```

```
  if e102055[i]=6 and -4<e99500[i]<0 then othpay5[i]=e99500[i];
  end;

/* if other compensation is corrected in the later part */
  do i=1 to 9;
  if e102259[i]=1 then do;

  /*report hourly wage at the beginning*/
  if e100239[i]=1 and e100254[i]>=0 then othpay5[i]=e100254[i];
  if e100239[i]=2 and e100254[i]>=0 and e100254e[i]>0 and e88000[i]>0 then
          othpay5[i]=e100254[i]*e100254e[i]/e88000[i];
  if e100239[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e100254[i]>=0 and e88000[i]>0
          then othpay5[i]=e100254[i]/e88000[i];
  if e100239[i]=4 and e100254[i]>=0 and e88000[i]>0 then othpay5[i]=e100254[i]/(2*e88000[i]);
  if e100239[i]=5 and e100254[i]>=0 and e88000[i]>0 then othpay5[i]=e100254[i]/(4.3*e88000[i]);
  if e100239[i]=6 and e100254[i]>=0 and e88000[i]>0 and e99500[i]>0 then
          othpay5[i]=e100254[i]/(e99500[i]*e88000[i]);
  if e100239[i]=8 and e100254[i]>=0 and e88000[i]>0 then othpay5[i]=e100254[i]/(2.15*e88000[i]);
  if e100239[i] in (9,14) then othpay5[i]=0;
   if e100239[i] in (12,13) and e100254[i]>=0 then do;
     /* condition based on the time unit to finish per othpay5. */
      if e100254c[i]=1 and e100254cb[i]>0 then othpay5[i]=(e100254[i]/e100254cb[i])*60;
      if e100254c[i]=2 and e100254cb[i]>0 then othpay5[i]=e100254[i]/e100254cb[i];
      if e100254c[i]=3 and e100254cb[i]>0 and e100254e[i]>=0 and e88000[i]>0 then
        othpay5[i]=(e100254[i]/e100254cb[i])*e100254e[i]/e88000[i];
      if e100254c[i]=4 and e100254cb[i]>0 and e88000[i]>0 then othpay5[i]=(e100254[i]/e100254cb[i])/e88000[i];
      if e100254c[i]=5 and e100254cb[i]>0 and e88000[i]>0 then
          othpay5[i]=(e100254[i]/e100254cb[i])/(e88000[i]*4.3);
    /* missing value */
      if -4<e100254c[i]<0 or -4<e100254cb[i]<=0 then othpay5[i]=-3;
      if e100254c[i] in (3,4,5) and -4<e88000[i]<=0 then othpay5[i]=-3;
      if e100254c[i]=3 and e100254e[i]<0 then othpay5[i]=-3;
  end;
   if e100239[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e100254[i]<0 then
          othpay5[i]=e100254[i];
   if e100239[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e88000[i]=0 then othpay5[i]=-3;
   if e100239[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e88000[i]<0 then
          othpay5[i]=e88000[i];
   if e100239[i]=2 and e100254e[i] le 0 then othpay5[i]=-3;
   if e100239[i]=2 and -4<e100254e[i]<0 then othpay5[i]=e100254e[i];

  /* non-hourly wage at the beginning, with overtime*/
  if e100239[i]=2 and e100254[i]>=0 and e100254e[i]>0 and e98429[i]>0 then
          othpay5[i]=e100254[i]*e100254e[i]/e98429[i];
  if e100239[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e100254[i]>=0 and e98429[i]>0
          then othpay5[i]=e100254[i]/e98429[i];
  if e100239[i]=4 and e100254[i]>=0 and e98429[i]>0 then othpay5[i]=e100254[i]/(2*e98429[i]);
  if e100239[i]=5 and e100254[i]>=0 and e98429[i]>0 then othpay5[i]=e100254[i]/(4.3*e98429[i]);
  if e100239[i]=6 and e100254[i]>=0 and e98429[i]>0 and e99500[i]>0 then
          othpay5[i]=e100254[i]/(e99500[i]*e98429[i]);
  if e100239[i]=8 and e100254[i]>=0 and e98429[i]>0 then othpay5[i]=e100254[i]/(2.15*e98429[i]);
  if e100239[i] in (12,13) and e100254[i]>=0 then do;
     /* condition based on the time unit to finish per othpay5. */
      if e100254c[i]=3 and e100254cb[i]>0 and e100254e[i]>=0 and e98429[i]>0 then
        othpay5[i]=(e100254[i]/e100254cb[i])*e100254e[i]/e98429[i];
      if e100254c[i]=4 and e100254cb[i]>0 and e98429[i]>0 then othpay5[i]=(e100254[i]/e100254cb[i])/e98429[i];
```

```
      if e100254c[i]=5 and e100254cb[i]>0 and e98429[i]>0 then
            othpay5[i]=(e100254[i]/e100254cb[i])/(e98429[i]*4.3);
    /* missing value */
      if e100254c[i] in (3,4,5) and -4<e98429[i]<=0 then othpay5[i]=-3;
  end;
   if e100239[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98429[i]=0 then othpay5[i]=-3;
   if e100239[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98429[i]<0 then
            othpay5[i]=e98429[i];

  /* non-hourly wage at the beginning, without overtime*/
  if e100239[i]=2 and e100254[i]>=0 and e100254e[i]>0 and e98402[i]>0 then
            othpay5[i]=e100254[i]*e100254e[i]/e98402[i];
  if e100239[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e100254[i]>=0 and e98402[i]>0
            then othpay5[i]=e100254[i]/e98402[i];
  if e100239[i]=4 and e100254[i]>=0 and e98402[i]>0 then othpay5[i]=e100254[i]/(2*e98402[i]);
  if e100239[i]=5 and e100254[i]>=0 and e98402[i]>0 then othpay5[i]=e100254[i]/(4.3*e98402[i]);
  if e100239[i]=6 and e100254[i]>=0 and e98402[i]>0 and e99500[i]>0 then
            othpay5[i]=e100254[i]/(e99500[i]*e98402[i]);
  if e100239[i]=8 and e100254[i]>=0 and e98402[i]>0 then othpay5[i]=e100254[i]/(2.15*e98402[i]);
   if e100239[i] in (12,13) and e100254[i]>=0 then do;
     /* condition based on the time unit to finish per othpay5. */
      if e100254c[i]=3 and e100254cb[i]>0 and e100254e[i]>=0 and e98402[i]>0 then
        othpay5[i]=(e100254[i]/e100254cb[i])*e100254e[i]/e98402[i];
      if e100254c[i]=4 and e100254cb[i]>0 and e98402[i]>0 then othpay5[i]=(e100254[i]/e100254cb[i])/e98402[i];
      if e100254c[i]=5 and e100254cb[i]>0 and e98402[i]>0 then
            othpay5[i]=(e100254[i]/e100254cb[i])/(e98402[i]*4.3);
    /* missing value */
      if e100254c[i] in (3,4,5) and -4<e98402[i]<=0 then othpay5[i]=-3;
  end;
   if e100239[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e98402[i]=0 then othpay5[i]=-3;
   if e100239[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e98402[i]<0 then
            othpay5[i]=e98402[i];
  if e100239[i]=6 and e99500[i] le 0 then othpay5[i]=-3;
  if e100239[i]=6 and -4<e99500[i]<0 then othpay5[i]=e99500[i];

 end;
 end;

/***************** overall start hourly compensation ****************************************/
array hrcomp hrcomp01 hrcomp02 hrcomp03 hrcomp04 hrcomp05 hrcomp06 hrcomp07 hrcomp08 hrcomp09;
do i=1 to 9;
  hrcomp[i]=0;
  if hrwg[i] ge 0 then hrcomp[i]=hrcomp[i]+hrwg[i];
  if ot[i] ge 0 then hrcomp[i]=hrcomp[i]+ot[i];
  if othpay1[i] ge 0 then hrcomp[i]=hrcomp[i]+othpay1[i];
  if othpay2[i] ge 0 then hrcomp[i]=hrcomp[i]+othpay2[i];
  if othpay3[i] ge 0 then hrcomp[i]=hrcomp[i]+othpay3[i];
  if othpay4[i] ge 0 then hrcomp[i]=hrcomp[i]+othpay4[i];
  if othpay5[i] ge 0 then hrcomp[i]=hrcomp[i]+othpay5[i];
  if -4<hrwg[i]<0 or -4<ot[i]<0 or -4<othpay1[i]<0 or -4<othpay2[i]<0 or -4<othpay3[i]<0 or -4<othpay4[i]<0 or -
            4<othpay5[i]<0 then hrcomp[i]=-3;
  if hrwg[i]=-4 then hrcomp[i]=-4;
end;


/************************************** end wages for youth ************************************/
```

```
******first, find the regular working hours for the people who have no overtime at the beginning;
array e3800ff e3800ff1-e3800ff9; /*which is the regular hours.*/

do i=1 to 9;
   if e38002[i] in (0,-4) or e3800f[i] in (0,-4) then e3800ff[i]=e3800f[i];
   if -4<e38002[i]<0 or -4<e3800f[i]<0 then e3800ff[i]=-3;
   if e38002[i]>0 and e3800f[i]>0 then e3800ff[i]=e3800f[i]-e38002[i];
end;

/*set up the end hourly wage for youths who report an hourly wage*/

do i=1 to 9;

/* end hourly wage for workers */
if (e37901b[i]=1 or e59900[i]=1) then do;

/* without overtime at the beginning*/
    if (e38013[i]=1 and e38014[i]=1 ) then do;
     if e38023[i]>=0 then hrwage[i]=e38023[i];
     /*missing value*/
     if -4<e38023[i]<0 then hrwage[i]=e38023[i];
    end;
/* with overtime at the beginning */
    if (e38106[i]=1 and e38107[i]=1) then do;
     if e38116[i]>=0 then hrwage[i]=e38116[i];
     /*missing value*/
     if -4<e38106[i]<0 then hrwage[i]=e38106[i];
    end;
end;

end;

/*set up the end hourly wage for youths who report their wage in daily units.*/

do i=1 to 9;

/* end hourly wage for workers */

if (e37901b[i]=1 or e59900[i]=1) then do;

/*without overtime at the beginning*/
    if (e38013[i]=1 and e38014[i]=2) then do;
     if e38023[i]>=0 and e3800b[i]=1 and e38027[i]>0 and e34402[i]>0 then daily[i]=e38023[i]*e38027[i]/e34402[i];
     if e38023[i]>=0 and e3800ff[i]>0 and e38027[i]>0 then daily[i]=e38023[i]*e38027[i]/e3800ff[i];
     /*missing value*/
     if e38023[i]>=0 and -4<e38027[i]<0 then daily[i]=e38027[i];
     if e38023[i]>=0 and e3800b[i]=1 and -4<e34402[i]<0 then daily[i]=e34402[i];
     if e38023[i]>=0 and -4<e3800ff[i]<0 then daily[i]=e3800ff[i];
     if e38023[i]>=0 and (e34402[i]=0 or e3800ff[i]=0) then daily[i]=-3;
    end;
/*with overtime at the beginning*/
    if (e38106[i]=1 and e38107[i]=2) then do;
     if e38116[i]>=0 and (e38102[i] ne 1 and e38102[i] ne 3) and e38116b[i]>0 and e34428[i]>0 then
          daily[i]=e38116b[i]*e38116[i]/e34428[i];
     if e38116[i]>=0 and (e38102[i]=1 or e38102[i]=3) and e38103[i]>0 and e38116b[i]>0 then
          daily[i]=e38116b[i]*e38116[i]/e38103[i];
     /*missing value*/
```

```
      if e38116[i]>=0 and -4<e38116b[i]<0 then daily[i]=e38116b[i];
      if e38116[i]>=0 and (e38102[i] ne 1 and e38102[i] ne 3) and -4<e34428[i]<0 then daily[i]=e34428[i];
      if e38116[i]>=0 and (e38102[i]=1 or e38102[i]=3) and -4<e38103[i]<0 then daily[i]=e38103[i];
      if e38116[i]>=0 and (e34428[i]=0 or e38103[i]=0) then daily[i]=-3;
     end;
end;

end;

/*set up the end hourly wage for youths who report their wage in weekly units.*/

do i=1 to 9;

/* end hourly wage for workers */

if (e37901b[i]=1 or e59900[i]=1) then do;

/*without overtime at the beginning */
     if e38013[i]=1 and e38014[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) then do;
      if e38023[i]>=0 and e3800b[i]=1 and e34402[i]>0 then weekly[i]=e38023[i]/e34402[i];
      if e38023[i]>=0 and e3800ff[i]>0 then weekly[i]=e38023[i]/e3800ff[i];
      /*missing value*/
      if e38023[i]>=0 and e3800b[i]=1 and -4<e34402[i]<0 then weekly[i]=e34402[i];
      if e38023[i]>=0 and -4<e3800ff[i]<0 then weekly[i]=e3800ff[i];
      if e38023[i]>=0 and (e34402[i]=0 or e3800ff[i]=0) then weekly[i]=-3;
       end;
/* with overtime at the beginning */
     if e38106[i]=1 and e38107[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) then do;
      if e38116[i]>=0 and (e38102[i] ne 1 and e38102[i] ne 3) and e34428[i]>0 then weekly[i]=e38116[i]/e34428[i];
      if e38116[i]>=0 and (e38102[i]=1 or e38102[i]=3) and e38103[i]>0 then weekly[i]=e38116[i]/e38103[i];
      /*missing value*/
      if e38116[i]>=0 and (e38102[i] ne 1 and e38102[i] ne 3) and -4<e34428[i]<0 then weekly[i]=e34428[i];
      if e38116[i]>=0 and (e38102[i]=1 or e38102[i]=3) and -4<e38103[i]<0 then weekly[i]=e38103[i];
      if e38116[i]>=0 and (e34428[i]=0 or e38103[i]=0) then weekly[i]=-3;
     end;
end;
end;

/*set up the end hourly wage for youths who report their wage in biwkly units.*/

do i=1 to 9;

/* end hourly wage for workers */

if (e37901b[i]=1 or e59900[i]=1) then do;

/* without overtime at the beginning */
     if (e38013[i]=1 and e38014[i]=4) then do;
      if e38023[i]>=0 and e3800b[i]=1 and e34402[i]>0 then biwkly[i]=e38023[i]/(2*e34402[i]);
      if e38023[i]>=0 and e3800ff[i]>0 then biwkly[i]=e38023[i]/(2*e3800ff[i]);
      /*missing value*/
      if e38023[i]>=0 and e3800b[i]=1 and -4<e34402[i]<0 then biwkly[i]=e34402[i];
      if e38023[i]>=0 and -4<e3800ff[i]<0 then biwkly[i]=e3800ff[i];
      if e38023[i]>=0 and (e34402[i]=0 or e3800ff[i]=0) then biwkly[i]=-3;
     end;
/* with overtime at the beginning*/
     if (e38106[i]=1 and e38107[i]=4) then do;
```

```
    if e38116[i]>=0 and (e38102[i] ne 1 and e38102[i] ne 3) and e34428[i]>0 then biwkly[i]=e38116[i]/(2*e34428[i]);
    if e38116[i]>=0 and (e38102[i]=1 or e38102[i]=3) and e38103[i]>0 then biwkly[i]=e38116[i]/(2*e38103[i]);
    /*missing value*/
    if e38116[i]>=0 and (e38102[i] ne 1 and e38102[i] ne 3) and -4<e34428[i]<0 then biwkly[i]=e34428[i];
    if e38116[i]>=0 and (e38102[i]=1 or e38102[i]=3) and -4<e38103[i]<0 then biwkly[i]=e38103[i];
    if e38116[i]>=0 and (e34428[i]=0 or e38103[i]=0) then biwkly[i]=-3;
   end;
end;

end;

/*set up the end hourly wage for youths who report their wage in month units.*/

do i=1 to 9;

/* end hourly wage for workers */

if (e37901b[i]=1 or e59900[i]=1) then do;

/* without overtime at the beginning */
   if (e38013[i]=1 and e38014[i]=5) then do;
    if e38023[i]>=0 and e3800b[i]=1 and e34402[i]>0 then month[i]=e38023[i]/(4.3*e34402[i]);
    if e38023[i]>=0 and e3800ff[i]>0 then month[i]=e38023[i]/(4.3*e3800ff[i]);
    /*missing value*/
    if e38023[i]>=0 and e3800b[i]=1 and -4<e34402[i]<0 then month[i]=e34402[i];
    if e38023[i]>=0 and -4<e3800ff[i]<0 then month[i]=e3800ff[i];
    if e38023[i]>=0 and (e34402[i]=0 or e3800ff[i]=0) then month[i]=-3;
   end;
/* with overtime at the beginning */
   if (e38106[i]=1 and e38107[i]=5) then do;
    if e38116[i]>=0 and (e38102[i] ne 1 and e38102[i] ne 3) and e34428[i]>0 then month[i]=e38116[i]/(4.3*e34428[i]);
    if e38116[i]>=0 and (e38102[i]=1 or e38102[i]=3) and e38103[i]>0 then month[i]=e38116[i]/(4.3*e38103[i]);
    /*missing value*/
    if e38116[i]>=0 and (e38102[i] ne 1 and e38102[i] ne 3) and -4<e34428[i]<0 then month[i]=e34428[i];
    if e38116[i]>=0 and (e38102[i]=1 or e38102[i]=3) and -4<e38103[i]<0 then month[i]=e38103[i];
    if e38116[i]>=0 and (e34428[i]=0 or e38103[i]=0) then month[i]=-3;
   end;
end;

end;

/*set up the end hourly wage for youths who report their wage in annual units.*/

do i=1 to 9;

/* end hourly wage for workers */

if (e37901b[i]=1 or e59900[i]=1) then do;

/* without overtime at the beginning */
   if (e38013[i]=1 and e38014[i]=6) then do;
    if e38023[i]>=0 and e3800b[i]=1 and e34402[i]>0 and e35600[i]>0 then annual[i]=e38023[i]/(e35600[i]*e34402[i]);
    if e38023[i]>=0 and e3800ff[i]>0 and e35600[i]>0 then annual[i]=e38023[i]/(e35600[i]*e3800ff[i]);
    /*missing value*/
    if e38023[i]>=0 and e3800b[i]=1 and -4<e34402[i]<0 then annual[i]=e34402[i];
    if e38023[i]>=0 and -4<e3800ff[i]<0 then annual[i]=e3800ff[i];
    if e38023[i]>=0 and (e34402[i]=0 or e3800ff[i]=0) then annual[i]=-3;
```

```
     if e35600[i] le 0 then annual[i]=-3;
     if -4<e35600[i]<0 then annual[i]=e35600[i];
    end;
/* with overtime at the beginning */
    if (e38106[i]=1 and e38107[i]=6) then do;
     if e38116[i]>=0 and (e38102[i] ne 1 and e38102[i] ne 3) and e34428[i]>0 and e35600[i]>0 then
            annual[i]=e38116[i]/(e35600[i]*e34428[i]);
     if e38116[i]>=0 and (e38102[i]=1 or e38102[i]=3) and e38103[i]>0 and e35600[i]>0 then
            annual[i]=e38116[i]/(e35600[i]*e38103[i]);
     /*missing value*/
     if e38116[i]>=0 and (e38102[i] ne 1 and e38102[i] ne 3) and -4<e34428[i]<0 then annual[i]=e34428[i];
     if e38116[i]>=0 and (e38102[i]=1 or e38102[i]=3) and -4<e38103[i]<0 then annual[i]=e38103[i];
     if e38116[i]>=0 and (e34428[i]=0 or e38103[i]=0) then annual[i]=-3;
     if e35600[i] le 0 then annual[i]=-3;
     if -4<e35600[i]<0 then annual[i]=e35600[i];
    end;
end;

end;

/*set up the end hourly wage for youths who report their wage in semim units.*/

do i=1 to 9;

/* end hourly wage for workers */

if (e37901b[i]=1 or e59900[i]=1) then do;

/* without overtime at the beginning*/
    if (e38013[i]=1 and e38014[i]=8) then do;
     if e38023[i]>=0 and e3800b[i]=1 and e34402[i]>0 then semim[i]=e38023[i]/(2.15*e34402[i]);
     if e38023[i]>=0 and e3800ff[i]>0 then semim[i]=e38023[i]/(2.15*e3800ff[i]);
     /*missing value*/
     if e38023[i]>=0 and e3800b[i]=1 and -4<e34402[i]<0 then semim[i]=e34402[i];
     if e38023[i]>=0 and -4<e3800ff[i]<0 then semim[i]=e3800ff[i];
     if e38023[i]>=0 and (e34402[i]=0 or e3800ff[i]=0) then semim[i]=-3;
    end;
/* with overtime at the beginning */
    if (e38106[i]=1 and e38107[i]=8) then do;
     if e38116[i]>=0 and (e38102[i] ne 1 and e38102[i] ne 3) and e34428[i]>0 then
            semim[i]=e38116[i]/(2.15*e34428[i]);
     if e38116[i]>=0 and (e38102[i]=1 or e38102[i]=3) and e38103[i]>0 then semim[i]=e38116[i]/(2.15*e38103[i]);
     /*missing value*/
     if e38116[i]>=0 and (e38102[i] ne 1 and e38102[i] ne 3) and -4<e34428[i]<0 then semim[i]=e34428[i];
     if e38116[i]>=0 and (e38102[i]=1 or e38102[i]=3) and -4<e38103[i]<0 then semim[i]=e38103[i];
     if e38116[i]>=0 and (e34428[i]=0 or e38103[i]=0) then semim[i]=-3;
    end;
end;

end;

/*set up the end hourly wage for youths who report their wage in per item.*/

do i=1 to 9;

/* end hourly wage for workers */
```

```
if (e37901b[i]=1 or e59900[i]=1) then do;

/* without overtime at the beginning */
    if (e38013[i]=1 and e38014[i] in (12,13)) then do;
    if e38023[i]>=0 then endhr1[i]=e3800ff[i];
     if e38023[i]>=0 and e3800b[i]=1  then endhr1[i]=e34402[i];
     /* condition based on the time unit to finish per item. */
       if e38024b[i]=1 and e38024c[i]>0 then item[i]=(e38023[i]/e38024c[i])*60;
       if e38024b[i]=2 and e38024c[i]>0 then item[i]=e38023[i]/e38024c[i];
       if e38024b[i]=3 and e38024c[i]>0 and e38027[i]>=0 and endhr1[i]>0 then
         item[i]=(e38023[i]/e38024c[i])*e38027[i]/endhr1[i];
       if e38024b[i]=4 and e38024c[i]>0 and endhr1[i]>0 then item[i]=(e38023[i]/e38024c[i])/endhr1[i];
       if e38024b[i]=5 and e38024c[i]>0 and endhr1[i]>0 then item[i]=(e38023[i]/e38024c[i])/(endhr1[i]*4.3);
     /* missing value */
       if -4<e38024b[i]<0 or -4<e38024c[i]<=0 then item[i]=-3;
       if e38024b[i] in (3,4,5) and -4<endhr1[i]<=0 then item[i]=-3;
       if e38024b[i]=3 and e38027[i]<0 then item[i]=-3;
    end;
/* with overtime at the beginning */
    if (e38106[i]=1 and e38107[i] in (12,13)) then do;
     if e38116[i]>=0 and (e38102[i] ne 1 and e38102[i] ne 3) then endhr2[i]=e34428[i];
     if e38116[i]>=0 and (e38102[i]=1 or e38102[i]=3) then endhr2[i]=e38103[i];
     /* condition based on the time unit to finish per item. */
       if e38116aaa[i]=1 and e38116aab[i]>0 then item[i]=(e38116[i]/e38116aab[i])*60;
       if e38116aaa[i]=2 and e38116aab[i]>0 then item[i]=e38116[i]/e38116aab[i];
       if e38116aaa[i]=3 and e38116aab[i]>0 and e38116b[i]>=0 and endhr2[i]>0 then
         item[i]=(e38116[i]/e38116aab[i])*e38116b[i]/endhr2[i];
       if e38116aaa[i]=4 and e38116aab[i]>0 and endhr2[i]>0 then item[i]=(e38116[i]/e38116aab[i])/endhr2[i];
       if e38116aaa[i]=5 and e38116aab[i]>0 and endhr2[i]>0 then item[i]=(e38116[i]/e38116aab[i])/(endhr2[i]*4.3);
     /* missing value */
       if -4<e38116aaa[i]<0 or -4<e38116aab[i]<=0 then item[i]=-3;
       if e38116aaa[i] in (3,4,5) and -4<endhr2[i]<=0 then item[i]=-3;
       if e38116aaa[i]=3 and e38116b[i]<0 then item[i]=-3;
    end;
end;

end;

/******** create the hourly rate of pay combining the information from end wage ***********/
do i=1 to 9;

if e37901b[i]=1 or e59900[i]=1 then do;

if annual[i] ge 0 then hrwg[i]=annual[i];
if month[i] ge 0 then hrwg[i]=month[i];
if biwkly[i] ge 0 then hrwg[i]=biwkly[i];
if weekly[i] ge 0 then hrwg[i]=weekly[i];
if daily[i] ge 0 then hrwg[i]=daily[i];
if hrwage[i] ge 0 then hrwg[i]=hrwage[i];
if semim[i] ge 0 then hrwg[i]=semim[i];
if item[i] ge 0 then hrwg[i]=item[i];
if hrwage[i] eq -1 or daily[i]=-1 or weekly[i] eq -1 or biwkly[i] eq -1 or month[i] eq -1 or
 annual[i] eq -1 or semim[i]=-1 or item[i]=-1 then hrwg[i]=-1;
if hrwage[i] eq -2 or daily[i]=-2 or weekly[i] eq -2 or biwkly[i] eq -2 or month[i] eq -2 or
 annual[i] eq -2 or semim[i]=-2 or item[i]=-2 then hrwg[i]=-2;
if hrwage[i] eq -3 or daily[i]=-3 or weekly[i] eq -3 or biwkly[i] eq -3 or month[i] eq -3 or
 annual[i] eq -3 or semim[i]=-3 or item[i]=-3 then hrwg[i]=-3;
```

```
end;
end;

/*set up the hourly wage for youths who report their wage in other manners */

do i=1 to 9.;

/* end hourly wage for workers */

if (e37901b[i]=1 or e59900[i]=1) then do;

/* if job was not offering any comp at the end */
    if (e38013[i]=1 and e38014[i] in (9,14)) then hrwg[i]=0;
    if e38013[i]=1 and -4<e38023[i]<0 then hrwg[i]=e38023[i];
/* if job was offering a comp at the end */
    if (e38106[i]=1 and e38107[i] in (9,14)) then hrwg[i]=0;
    if e38106[i]=1 and -4<e38116[i]<0 then hrwg[i]=e38116[i];

end;
end;

/* report -1,-2 or -3 if the amount reported is -1, -2 or -3*/
do i=1 to 9;
  if e37901b[i]=1 or e59900[i]=1 then do;
    if -4<e38023[i]<0 then hrwg[i]=e38023[i];
    if -4<e38116[i]<0 then hrwg[i]=e38116[i];
  end;
end;

/* the end wage if job lasts for >=13 weeks and report the same amount but diff. hours*/

/* note: for this case, we only change hourly rate hrwg[i] without changing hrwage[i], daily[i], weeklw[i] biwkly[i],
          month[i], annual[i] or semim[i]*/

do i=1 to 9;
   if ((e37901b[i]=1 or e59900[i]=1) and e38013[i]=0 and e3800b[i]=0) and hrwg[i]>=0
      and e34402[i]>0 and e3800ff[i]>0 then hrwg[i]=hrwg[i]*e34402[i]/e3800ff[i];
   if ((e37901b[i]=1 or e59900[i]=1) and e38013[i]=0 and e3800b[i]=0) and e3800ff[i]=0 then hrwg[i]=-3;
   if ((e37901b[i]=1 or e59900[i]=1) and e38013[i]=0 and e3800b[i]=0) and -4<e3800ff[i]<0 then hrwg[i]=e3800ff[i];
   if ((e37901b[i]=1 or e59900[i]=1) and e38106[i]=0 and (e38102[i]=1 or e38102[i]=3))
      and hrwg[i]>=0 and e34428[i]>0 and e38103[i]>0 then hrwg[i]=hrwg[i]*e34428[i]/e38103[i];
   if ((e37901b[i]=1 or e59900[i]=1) and e38106[i]=0 and (e38102[i]=1 or e38102[i]=3))
      and hrwg[i]>=0 and e34402[i]>0 and e38103[i]>0 then hrwg[i]=hrwg[i]*e34402[i]/e38103[i];
   if ((e37901b[i]=1 or e59900[i]=1) and e38106[i]=0 and (e38102[i]=1 or e38102[i]=3))
      and e38103[i]=0 then hrwg[i]=-3;
   if ((e37901b[i]=1 or e59900[i]=1) and e38106[i]=0 and (e38102[i]=1 or e38102[i]=3))
      and -4<e38103[i]<0 then hrwg[i]=e38103[i];
end;

/************************ add the end compensation ******************************/
/************************ amount paid of overtime ****************************/

do i=1 to 9;

if (e37901b[i]=1 or e59900[i]=1) then do;
```

```
/**** without compensation at the beginning *****/

if e38001[i]=0 then ot[i]=0;

  else do;
  if e38003[i]=0 then multot=multot+1;
  if e38002[i]>0 and e38003[i]=1 and e38012[i]>=0 then ot[i]=e38012[i];
  if e38002[i]>0 and e38003[i]=2 and e38012[i]>=0 and e38012b[i]>0 then ot[i]=e38012[i]*e38012b[i]/e38002[i];
  if e38002[i]>0 and e38003[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38012[i]>=0 then
          ot[i]=e38012[i]/e38002[i];
  if e38002[i]>0 and e38003[i]=4 and e38012[i]>=0 then ot[i]=e38012[i]/(2*e38002[i]);
  if e38002[i]>0 and e38003[i]=5 and e38012[i]>=0 then ot[i]=e38012[i]/(4.3*e38002[i]);
  if e38002[i]>0 and e38003[i]=6 and e38012[i]>=0 then ot[i]=-3; /* since no weeks per year aavailable. */
  if e38002[i]>0 and e38003[i]=8 and e38012[i]>=0 then ot[i]=e38012[i]/(2.15*e38002[i]);
  if e38003[i] in (9,14) then ot[i]=0;
  if e38003[i] in (12,13) and e38012[i]>=o then do;
    /* condition based on the time unit to finish per ot. */
      if e38012aaa[i]=1 and e38012aab[i]>0 then ot[i]=(e38012[i]/e38012aab[i])*60;
      if e38012aaa[i]=2 and e38012aab[i]>0 then ot[i]=e38012[i]/e38012aab[i];
      if e38012aaa[i]=3 and e38012aab[i]>0 and e38012b[i]>=0 and e38002[i]>0 then
        ot[i]=(e38012[i]/e38012aab[i])*e38012b[i]/e38002[i];
      if e38012aaa[i]=4 and e38012aab[i]>0 and e38002[i]>0 then ot[i]=(e38012[i]/e38012aab[i])/e38002[i];
      if e38012aaa[i]=5 and e38012aab[i]>0 and e38002[i]>0 then ot[i]=(e38012[i]/e38012aab[i])/(e38002[i]*4.3);
    /* missing value */
      if -4<e38012aaa[i]<0 or -4<e38012aab[i]<=0 then ot[i]=-3;
      if e38012aaa[i] in (3,4,5) and -4<e38002[i]<=0 then ot[i]=-3;
      if e38012aaa[i]=3 and e38012b[i]<0 then ot[i]=-3;
  end;

  /*missing value*/
  if -4<e38002[i]<0 then ot[i]=e38002[i];
  if e38012[i]>=0 and e38002[i]=0 then ot[i]=-3;
  if -4<e38012[i]<0 then ot[i]=e38012[i];
   if e38003[i]=2 and e38012b[i] le 0 then ot[i]=-3;
   if e38003[i]=2 and -4<e38012b[i]<0 then ot[i]=e38012b[i];
  end;

/****  with compensation at the beginning  *****/

if e38201[i]=1 then do;
  if e38202[i]=0 then multot=multot+1;
  if e38101[i]=1 then do;       /*same no. of hours as at the beginning */

  if e19200[i]=1 then do;       /* report hourly rate of pay at the beginning*/

    if e24501[i]>0 and e38202[i]=1  and e38211[i]>=0 then ot[i]=e38211[i];
    if e24501[i]>0 and e38202[i]=2  and e38211b[i]>0 and e38211[i]>=0 then ot[i]=e38211[i]*e38211b[i]/e24501[i];
    if e24501[i]>0 and e38202[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2)  and e38211[i]>=0
          then ot[i]=e38211[i]/e24501[i];
    if e24501[i]>0 and e38202[i]=4  and e38211[i]>=0 then ot[i]=e38211[i]/(2*e24501[i]);
    if e24501[i]>0 and e38202[i]=5  and e38211[i]>=0 then ot[i]=e38211[i]/(4.3*e24501[i]);
    if e24501[i]>0 and e38202[i]=6  and e38211[i]>=0 then ot[i]=-3; /* since no weeks per year available. */
    if e24501[i]>0 and e38202[i]=8  and e38211[i]>=0 then ot[i]=e38211[i]/(2.15*e24501[i]);
    if e38202[i] in (9,14) then ot[i]=0;
    if e38202[i] in (12,13) and e38211[i]>=o then do;
    /* condition based on the time unit to finish per ot. */
      if e38211aaa[i]=1 and e38211aab[i]>0 then ot[i]=(e38211[i]/e38211aab[i])*60;
```

```
      if e38211aaa[i]=2 and e38211aab[i]>0 then ot[i]=e38211[i]/e38211aab[i];
      if e38211aaa[i]=3 and e38211aab[i]>0 and e38211b[i]>=0 and e24501[i]>0 then
        ot[i]=(e38211[i]/e38211aab[i])*e38211b[i]/e24501[i];
      if e38211aaa[i]=4 and e38211aab[i]>0 and e24501[i]>0 then ot[i]=(e38211[i]/e38211aab[i])/e24501[i];
      if e38211aaa[i]=5 and e38211aab[i]>0 and e24501[i]>0 then ot[i]=(e38211[i]/e38211aab[i])/(e24501[i]*4.3);
    /* missing value */
      if -4<e38211aaa[i]<0 or -4<e38211aab[i]<=0 then ot[i]=-3;
      if e38211aaa[i] in (3,4,5) and -4<e24501[i]<=0 then ot[i]=-3;
      if e38211aaa[i]=3 and e38211b[i]<0 then ot[i]=-3;
   end;

 if -4<e24501[i]<0 then ot[i]=e24501[i];
 if e38211[i]>=0 and e24501[i]=0 then ot[i]=-3;
 if -4<e38211[i]<0 then ot[i]=e38211[i];
  if e38202[i]=2 and e38211b[i] le 0 then ot[i]=-3;
  if e38202[i]=2 and -4<e38211b[i]<0 then ot[i]=e38211b[i];

 end;

 if e19200[i] ne 1 then do;     /*report payment in other units at the begining*/
    if e34403[i]>0 and e38202[i]=1  and e38211[i]>=0 then ot[i]=e38211[i];
    if e34403[i]>0 and e38202[i]=2  and e38211b[i]>0 and e38211[i]>=0 then ot[i]=e38211[i]*e38211b[i]/e34403[i];
    if e34403[i]>0 and e38202[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2)  and e38211[i]>=0
           then ot[i]=e38211[i]/e34403[i];
    if e34403[i]>0 and e38202[i]=4  and e38211[i]>=0 then ot[i]=e38211[i]/(2*e34403[i]);
    if e34403[i]>0 and e38202[i]=5  and e38211[i]>=0 then ot[i]=e38211[i]/(4.3*e34403[i]);
    if e34403[i]>0 and e38202[i]=6  and e38211[i]>=0 then ot[i]=-3; /* since no weeks per year available. */
    if e34403[i]>0 and e38202[i]=8  and e38211[i]>=0 then ot[i]=e38211[i]/(2.15*e34403[i]);
    if e38202[i] in (9,14) then ot[i]=0;
    if e38202[i] in (12,13) and e38211[i]>=o then do;
    /* condition based on the time unit to finish per ot. */
      if e38211aaa[i]=1 and e38211aab[i]>0 then ot[i]=(e38211[i]/e38211aab[i])*60;
      if e38211aaa[i]=2 and e38211aab[i]>0 then ot[i]=e38211[i]/e38211aab[i];
      if e38211aaa[i]=3 and e38211aab[i]>0 and e38211b[i]>=0 and e34403[i]>0 then
        ot[i]=(e38211[i]/e38211aab[i])*e38211b[i]/e34403[i];
      if e38211aaa[i]=4 and e38211aab[i]>0 and e34403[i]>0 then ot[i]=(e38211[i]/e38211aab[i])/e34403[i];
      if e38211aaa[i]=5 and e38211aab[i]>0 and e34403[i]>0 then ot[i]=(e38211[i]/e38211aab[i])/(e34403[i]*4.3);
    /* missing value */
      if -4<e38211aaa[i]<0 or -4<e38211aab[i]<=0 then ot[i]=-3;
      if e38211aaa[i] in (3,4,5) and -4<e34403[i]<=0 then ot[i]=-3;
      if e38211aaa[i]=3 and e38211b[i]<0 then ot[i]=-3;
   end;

 if -4<e34403[i]<0 then ot[i]=e34403[i];
 if e38211[i]>=0 and e34403[i]=0 then ot[i]=-3;
 if -4<e38211[i]<0 then ot[i]=e38211[i];
  if e38202[i]=2 and e38211b[i] le 0 then ot[i]=-3;
  if e38202[i]=2 and -4<e38211b[i]<0 then ot[i]=e38211b[i];

 end;

end;

  else do;      /* different no. of hours from the beginning*/

    if e38105[i]>0 and e38202[i]=1  and e38211[i]>=0 then ot[i]=e38211[i];
    if e38105[i]>0 and e38202[i]=2  and e38211b[i]>0 and e38211[i]>=0 then ot[i]=e38211[i]*e38211b[i]/e38105[i];
```

```
    if e38105[i]>0 and e38202[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2)  and e38211[i]>=0
          then ot[i]=e38211[i]/e38105[i];
    if e38105[i]>0 and e38202[i]=4  and e38211[i]>=0 then ot[i]=e38211[i]/(2*e38105[i]);
    if e38105[i]>0 and e38202[i]=5  and e38211[i]>=0 then ot[i]=e38211[i]/(4.3*e38105[i]);
    if e38105[i]>0 and e38202[i]=6  and e38211[i]>=0 then ot[i]=-3; /* snice no weeks per year available. */
    if e38105[i]>0 and e38202[i]=8  and e38211[i]>=0 then ot[i]=e38211[i]/(2.15*e38105[i]);
    if e38202[i] in (9,14) then ot[i]=0;
    if e38202[i] in (12,13) and e38211[i]>=o then do;
    /* condition based on the time unit to finish per ot. */
      if e38211aaa[i]=1 and e38211aab[i]>0 then ot[i]=(e38211[i]/e38211aab[i])*60;
      if e38211aaa[i]=2 and e38211aab[i]>0 then ot[i]=e38211[i]/e38211aab[i];
      if e38211aaa[i]=3 and e38211aab[i]>0 and e38211b[i]>=0 and e38211[i]>0 then
        ot[i]=(e38211[i]/e38211aab[i])*e38211b[i]/e38105[i];
      if e38211aaa[i]=4 and e38211aab[i]>0 and e38211[i]>0 then ot[i]=(e38211[i]/e38211aab[i])/e38105[i];
      if e38211aaa[i]=5 and e38211aab[i]>0 and e38211[i]>0 then ot[i]=(e38211[i]/e38211aab[i])/(e38105[i]*4.3);
    /* missing value */
      if -4<e38211aaa[i]<0 or -4<e38211aab[i]<=0 then ot[i]=-3;
      if e38211aaa[i] in (3,4,5) and -4<e38211[i]<=0 then ot[i]=-3;
      if e38211aaa[i]=3 and e38211b[i]<0 then ot[i]=-3;
   end;

  if -4<e38105[i]<0 then ot[i]=e38105[i];
  if e38211[i]>=0 and e38105[i]=0 then ot[i]=-3;
  if -4<e38211[i]<0 then ot[i]=e38211[i];
   if e38202[i]=2 and e38211b[i] le 0 then ot[i]=-3;
   if e38202[i]=2 and -4<e38211b[i]<0 then ot[i]=e38211b[i];

  end;

 end;
end;
end;

/******************** non-overtime payment  ****************************/
do i=1 to 9;

if (e37901b[i]=1 or e59900[i]=1) then do;

/*** case i. without overtime at the beginning, same no. of hours. ***/
if e3800b[i]=1 and e212001[i] ne 1 then do;

 /* report hourly wage at the beginning*/
  /* for tips*/

  if e23901[i]>0 and e384071[i]=1  and e384161[i]>=0 then othpay1[i]=e384161[i];
  if e23901[i]>0 and e384071[i]=2  and e38416k1[i]>0 and e384161[i]>=0 then
          othpay1[i]=(e384161[i]*e38416k1[i])/e23901[i];
  if e23901[i]>0 and e384071[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2)  and e384161[i]>=0
          then othpay1[i]=e384161[i]/e23901[i];
  if e23901[i]>0 and e384071[i]=4  and e384161[i]>=0 then othpay1[i]=e384161[i]/(2*e23901[i]);
  if e23901[i]>0 and e384071[i]=5  and e384161[i]>=0 then othpay1[i]=e384161[i]/(4.3*e23901[i]);
  if e23901[i]>0 and e384071[i]=6  and e384161[i]>=0 and e35600[i]>0 then
          othpay1[i]=e384161[i]/(e35600[i]*e23901[i]);
  if e23901[i]>0 and e384071[i]=8  and e384161[i]>=0 then othpay1[i]=e384161[i]/(2.15*e23901[i]);
  if e384071[i] in (9,14) then othpay1[i]=0;
  if e384071[i] in (12,13) and e384161[i]>=o then do;
     if e38416g1[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
```

```
        if e225301[i] in (1,2,4,5) and e225001[i]>0 and othpay1[i]>=0 then
            othpay1[i]=othpay1[i]/e225001[i]*e384161[i];
        if e225301[i]=3 and e225001[i]>0 and e38416k1[i]>0 and e225501[i]>0 and othpay1[i]>=0
           then othpay1[i]=othpay1[i]/(e225001[i]*e225501[i])*e384161[i]*e38416k1[i];
        if e225301[i]=3 and e38416k1[i]<=0 then othpay1[i]=-3;
      end;
      else do; /*in all the other cases */
       if e38416i1[i]=1 and e38416ib1[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])*60;
       if e38416i1[i]=2 and e38416ib1[i]>0 then othpay1[i]=e384161[i]/e38416ib1[i];
       if e38416i1[i]=3 and e38416ib1[i]>0 and e38416k1[i]>=0 and e23901[i]>0 then
         othpay1[i]=(e384161[i]/e38416ib1[i])*e38416k1[i]/e23901[i];
       if e38416i1[i]=4 and e38416ib1[i]>0 and e23901[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])/e23901[i];
       if e38416i1[i]=5 and e38416ib1[i]>0 and e23901[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])/(e23901[i]*4.3);
     /* missing value */
       if -4<e38416i1[i]<0 or -4<e38416ib1[i]<=0 then othpay1[i]=-3;
       if e38416i1[i] in (3,4,5) and -4<e23901[i]<=0 then othpay1[i]=-3;
       if e38416i1[i]=3 and e38416k1[i]<0 then othpay1[i]=-3;
      end;
    end;

   if e384071[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384161[i]<0 then
            othpay1[i]=e225001[i];
   if e384071[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay1[i]=-3;
   if e384071[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
            othpay1[i]=e23901[i];
   if e384071[i]=2 and e38416k1[i] le 0 then othpay1[i]=-3;
   if e384071[i]=2 and -4<e38416k1[i]<0 then othpay1[i]=e225501[i];
   if e384071[i]=6 and e35600[i] le 0 then othpay1[i]=-3;
   if e384071[i]=6 and -4<e35600[i]<0 then othpay1[i]=e35600[i];

/*for commissions*/

   if e23901[i]>0 and e384072[i]=1 and e384162[i]>=0 then othpay2[i]=e384162[i];
   if e23901[i]>0 and e384072[i]=2 and e38416k2[i]>0 and e384162[i]>=0 then
            othpay2[i]=(e384162[i]*e38416k2[i])/e23901[i];
   if e23901[i]>0 and e384072[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384162[i]>=0
            then othpay2[i]=e384162[i]/e23901[i];
   if e23901[i]>0 and e384072[i]=4 and e384162[i]>=0 then othpay2[i]=e384162[i]/(2*e23901[i]);
   if e23901[i]>0 and e384072[i]=5 and e384162[i]>=0 then othpay2[i]=e384162[i]/(4.3*e23901[i]);
   if e23901[i]>0 and e384072[i]=6 and e384162[i]>=0 and e35600[i]>0 then
            othpay2[i]=e384162[i]/(e35600[i]*e23901[i]);
   if e23901[i]>0 and e384072[i]=8 and e384162[i]>=0 then othpay2[i]=e384162[i]/(2.15*e23901[i]);
   if e384072[i] in (9,14) then othpay2[i]=0;
   if e384072[i] in (12,13) and e384162[i]>=o then do;
      if e38416g2[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
        if e225302[i] in (1,2,4,5) and e225002[i]>0 and othpay2[i]>=0 then
            othpay2[i]=othpay2[i]/e225002[i]*e384162[i];
        if e225302[i]=3 and e225002[i]>0 and e38416k2[i]>0 and e225502[i]>0 and othpay2[i]>=0
           then othpay2[i]=othpay2[i]/(e225002[i]*e225502[i])*e384162[i]*e38416k2[i];
        if e225302[i]=3 and e38416k2[i]<=0 then othpay2[i]=-3;
      end;
      else do; /*in all the other cases */
       if e38416i2[i]=1 and e38416ib2[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])*60;
       if e38416i2[i]=2 and e38416ib2[i]>0 then othpay2[i]=e384162[i]/e38416ib2[i];
       if e38416i2[i]=3 and e38416ib2[i]>0 and e38416k2[i]>=0 and e23901[i]>0 then
         othpay2[i]=(e384162[i]/e38416ib2[i])*e38416k2[i]/e23901[i];
       if e38416i2[i]=4 and e38416ib2[i]>0 and e23901[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])/e23901[i];
```

```
      if e38416i2[i]=5 and e38416ib2[i]>0 and e23901[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])/(e23901[i]*4.3);
    /* missing value */
      if -4<e38416i2[i]<0 or -4<e38416ib2[i]<=0 then othpay2[i]=-3;
      if e38416i2[i] in (3,4,5) and -4<e23901[i]<=0 then othpay2[i]=-3;
      if e38416i2[i]=3 and e38416k2[i]<0 then othpay2[i]=-3;
     end;
   end;

  if e384072[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384162[i]<0 then
          othpay2[i]=e225002[i];
  if e384072[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay2[i]=-3;
  if e384072[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
          othpay2[i]=e23901[i];
  if e384072[i]=2 and e38416k2[i] le 0 then othpay2[i]=-3;
  if e384072[i]=2 and -4<e38416k2[i]<0 then othpay2[i]=e225502[i];
  if e384072[i]=6 and e35600[i] le 0 then othpay2[i]=-3;
  if e384072[i]=6 and -4<e35600[i]<0 then othpay2[i]=e35600[i];

/*for bonuses*/

  if e23901[i]>0 and e384073[i]=1 and e384163[i]>=0 then othpay3[i]=e384163[i];
  if e23901[i]>0 and e384073[i]=2 and e38416k3[i]>0 and e384163[i]>=0 then
          othpay3[i]=(e384163[i]*e38416k3[i])/e23901[i];
  if e23901[i]>0 and e384073[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384163[i]>=0
          then othpay3[i]=e384163[i]/e23901[i];
  if e23901[i]>0 and e384073[i]=4 and e384163[i]>=0 then othpay3[i]=e384163[i]/(2*e23901[i]);
  if e23901[i]>0 and e384073[i]=5 and e384163[i]>=0 then othpay3[i]=e384163[i]/(4.3*e23901[i]);
  if e23901[i]>0 and e384073[i]=6 and e384163[i]>=0 and e35600[i]>0 then
          othpay3[i]=e384163[i]/(e35600[i]*e23901[i]);
  if e23901[i]>0 and e384073[i]=8 and e384163[i]>=0 then othpay3[i]=e384163[i]/(2.15*e23901[i]);
  if e384073[i] in (9,14) then othpay3[i]=0;
  if e384073[i] in (12,13) and e384163[i]>=o then do;
     if e38416g3[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
       if e225303[i] in (1,2,4,5) and e225003[i]>0 and othpay3[i]>=0 then
          othpay3[i]=othpay3[i]/e225003[i]*e384163[i];
       if e225303[i]=3 and e225003[i]>0 and e38416k3[i]>0 and e225503[i]>0 and othpay3[i]>=0
         then othpay3[i]=othpay3[i]/(e225003[i]*e225503[i])*e384163[i]*e38416k3[i];
       if e225303[i]=3 and e38416k3[i]<=0 then othpay3[i]=-3;
     end;
     else do; /*in all the other cases */
      if e38416i3[i]=1 and e38416ib3[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])*60;
      if e38416i3[i]=2 and e38416ib3[i]>0 then othpay3[i]=e384163[i]/e38416ib3[i];
      if e38416i3[i]=3 and e38416ib3[i]>0 and e38416k3[i]>=0 and e23901[i]>0 then
        othpay3[i]=(e384163[i]/e38416ib3[i])*e38416k3[i]/e23901[i];
      if e38416i3[i]=4 and e38416ib3[i]>0 and e23901[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])/e23901[i];
      if e38416i3[i]=5 and e38416ib3[i]>0 and e23901[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])/(e23901[i]*4.3);
    /* missing value */
      if -4<e38416i3[i]<0 or -4<e38416ib3[i]<=0 then othpay3[i]=-3;
      if e38416i3[i] in (3,4,5) and -4<e23901[i]<=0 then othpay3[i]=-3;
      if e38416i3[i]=3 and e38416k3[i]<0 then othpay3[i]=-3;
     end;
   end;

  if e384073[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384163[i]<0 then
          othpay3[i]=e225003[i];
  if e384073[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay3[i]=-3;
```

```
  if e384073[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
          othpay3[i]=e23901[i];
  if e384073[i]=2 and e38416k3[i] le 0 then othpay3[i]=-3;
  if e384073[i]=2 and -4<e38416k3[i]<0 then othpay3[i]=e225503[i];
  if e384073[i]=6 and e35600[i] le 0 then othpay3[i]=-3;
  if e384073[i]=6 and -4<e35600[i]<0 then othpay3[i]=e35600[i];

/*for incentive pay*/

  if e23901[i]>0 and e384074[i]=1  and e384164[i]>=0 then othpay4[i]=e384164[i];
  if e23901[i]>0 and e384074[i]=2  and e38416k4[i]>0 and e384164[i]>=0 then
          othpay4[i]=(e384164[i]*e38416k4[i])/e23901[i];
  if e23901[i]>0 and e384074[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2)  and e384164[i]>=0
          then othpay4[i]=e384164[i]/e23901[i];
  if e23901[i]>0 and e384074[i]=4  and e384164[i]>=0 then othpay4[i]=e384164[i]/(2*e23901[i]);
  if e23901[i]>0 and e384074[i]=5  and e384164[i]>=0 then othpay4[i]=e384164[i]/(4.3*e23901[i]);
  if e23901[i]>0 and e384074[i]=6  and e384164[i]>=0 and e35600[i]>0 then
          othpay4[i]=e384164[i]/(e35600[i]*e23901[i]);
  if e23901[i]>0 and e384074[i]=8  and e384164[i]>=0 then othpay4[i]=e384164[i]/(2.15*e23901[i]);
  if e384074[i] in (9,14) then othpay4[i]=0;
  if e384074[i] in (12,13) and e384164[i]>=o then do;
     if e38416g4[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
        if e225304[i] in (1,2,4,5) and e225004[i]>0 and othpay4[i]>=0 then
           othpay4[i]=othpay4[i]/e225004[i]*e384164[i];
        if e225304[i]=3 and e225004[i]>0 and e38416k4[i]>0 and e225504[i]>0 and othpay4[i]>=0
          then othpay4[i]=othpay4[i]/(e225004[i]*e225504[i])*e384164[i]*e38416k4[i];
        if e225304[i]=3 and e38416k4[i]<=0 then othpay4[i]=-3;
     end;
     else do; /*in all the other cases */
      if e38416i4[i]=1 and e38416ib4[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])*60;
      if e38416i4[i]=2 and e38416ib4[i]>0 then othpay4[i]=e384164[i]/e38416ib4[i];
      if e38416i4[i]=3 and e38416ib4[i]>0 and e38416k4[i]>=0 and e23901[i]>0 then
        othpay4[i]=(e384164[i]/e38416ib4[i])*e38416k4[i]/e23901[i];
      if e38416i4[i]=4 and e38416ib4[i]>0 and e23901[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])/e23901[i];
      if e38416i4[i]=5 and e38416ib4[i]>0 and e23901[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])/(e23901[i]*4.3);
    /* missing value */
      if -4<e38416i4[i]<0 or -4<e38416ib4[i]<=0 then othpay4[i]=-3;
      if e38416i4[i] in (3,4,5) and -4<e23901[i]<=0 then othpay4[i]=-3;
      if e38416i4[i]=3 and e38416k4[i]<0 then othpay4[i]=-3;
     end;
   end;

  if e384074[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384164[i]<0 then
          othpay4[i]=e225004[i];
  if e384074[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay4[i]=-3;
  if e384074[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
          othpay4[i]=e23901[i];
  if e384074[i]=2 and e38416k4[i] le 0 then othpay4[i]=-3;
  if e384074[i]=2 and -4<e38416k4[i]<0 then othpay4[i]=e225504[i];
  if e384074[i]=6 and e35600[i] le 0 then othpay4[i]=-3;
  if e384074[i]=6 and -4<e35600[i]<0 then othpay4[i]=e35600[i];

/*for others*/

  if e23901[i]>0 and e384075[i]=1 and e384165[i]>=0 then othpay5[i]=e384165[i];
  if e23901[i]>0 and e384075[i]=2 and e38416k5[i]>0 and e384165[i]>=0 then
          othpay5[i]=(e384165[i]*e38416k5[i])/e23901[i];
```

```
  if e23901[i]>0 and e384075[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384165[i]>=0
          then othpay5[i]=e384165[i]/e23901[i];
  if e23901[i]>0 and e384075[i]=4 and e384165[i]>=0 then othpay5[i]=e384165[i]/(2*e23901[i]);
  if e23901[i]>0 and e384075[i]=5 and e384165[i]>=0 then othpay5[i]=e384165[i]/(4.3*e23901[i]);
  if e23901[i]>0 and e384075[i]=6 and e384165[i]>=0 and e35600[i]>0 then
          othpay5[i]=e384165[i]/(e35600[i]*e23901[i]);
  if e23901[i]>0 and e384075[i]=8 and e384165[i]>=0 then othpay5[i]=e384165[i]/(2.15*e23901[i]);
  if e384075[i] in (9,14) then othpay5[i]=0;
  if e384075[i] in (12,13) and e384165[i]>=o then do;
     if e38416g5[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
       if e225305[i] in (1,2,4,5) and e225005[i]>0 and othpay5[i]>=0 then
           othpay5[i]=othpay5[i]/e225005[i]*e384165[i];
       if e225305[i]=3 and e225005[i]>0 and e38416k5[i]>0 and e225505[i]>0 and othpay5[i]>=0
          then othpay5[i]=othpay5[i]/(e225005[i]*e225505[i])*e384165[i]*e38416k5[i];
       if e225305[i]=3 and e38416k5[i]<=0 then othpay5[i]=-3;
     end;
     else do; /*in all the other cases */
      if e38416i5[i]=1 and e38416ib5[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])*60;
      if e38416i5[i]=2 and e38416ib5[i]>0 then othpay5[i]=e384165[i]/e38416ib5[i];
      if e38416i5[i]=3 and e38416ib5[i]>0 and e38416k5[i]>=0 and e23901[i]>0 then
        othpay5[i]=(e384165[i]/e38416ib5[i])*e38416k5[i]/e23901[i];
      if e38416i5[i]=4 and e38416ib5[i]>0 and e23901[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])/e23901[i];
      if e38416i5[i]=5 and e38416ib5[i]>0 and e23901[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])/(e23901[i]*4.3);
     /* missing value */
      if -4<e38416i5[i]<0 or -4<e38416ib5[i]<=0 then othpay5[i]=-3;
      if e38416i5[i] in (3,4,5) and -4<e23901[i]<=0 then othpay5[i]=-3;
      if e38416i5[i]=3 and e38416k5[i]<0 then othpay5[i]=-3;
     end;
   end;

  if e384075[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384165[i]<0 then
          othpay5[i]=e225005[i];
  if e384075[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay5[i]=-3;
  if e384075[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
          othpay5[i]=e23901[i];
  if e384075[i]=2 and e38416k5[i] le 0 then othpay5[i]=-3;
  if e384075[i]=2 and -4<e38416k5[i]<0 then othpay5[i]=e225505[i];
  if e384075[i]=6 and e35600[i] le 0 then othpay5[i]=-3;
  if e384075[i]=6 and -4<e35600[i]<0 then othpay5[i]=e35600[i];

/* report nonhourly wage at the beginning*/

 /* for tips*/

  if e34402[i]>0 and e384071[i]=1 and e384161[i]>=0 then othpay1[i]=e384161[i];
  if e34402[i]>0 and e384071[i]=2 and e38416k1[i]>0 and e384161[i]>=0 then
          othpay1[i]=(e384161[i]*e38416k1[i])/e34402[i];
  if e34402[i]>0 and e384071[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384161[i]>=0
          then othpay1[i]=e384161[i]/e34402[i];
  if e34402[i]>0 and e384071[i]=4 and e384161[i]>=0 then othpay1[i]=e384161[i]/(2*e34402[i]);
  if e34402[i]>0 and e384071[i]=5 and e384161[i]>=0 then othpay1[i]=e384161[i]/(4.3*e34402[i]);
  if e34402[i]>0 and e384071[i]=6 and e384161[i]>=0 and e35600[i]>0 then
          othpay1[i]=e384161[i]/(e35600[i]*e34402[i]);
  if e34402[i]>0 and e384071[i]=8 and e384161[i]>=0 then othpay1[i]=e384161[i]/(2.15*e34402[i]);
  if e384071[i] in (12,13) and e384161[i]>=o then do;
      if e38416i1[i]=3 and e38416ib1[i]>0 and e38416k1[i]>=0 and e34402[i]>0 then
        othpay1[i]=(e384161[i]/e38416ib1[i])*e38416k1[i]/e34402[i];
```

```
      if e38416i1[i]=4 and e38416ib1[i]>0 and e34402[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])/e34402[i];
      if e38416i1[i]=5 and e38416ib1[i]>0 and e34402[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])/(e34402[i]*4.3);
    /* missing value */
      if e38416i1[i] in (3,4,5) and -4<e34402[i]<=0 then othpay1[i]=-3;
  end;

  if e384071[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay1[i]=-3;
  if e384071[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          othpay1[i]=e34402[i];

/*for commissions*/

  if e34402[i]>0 and e384072[i]=1 and e384162[i]>=0 then othpay2[i]=e384162[i];
  if e34402[i]>0 and e384072[i]=2 and e38416k2[i]>0 and e384162[i]>=0 then
          othpay2[i]=(e384162[i]*e38416k2[i])/e34402[i];
  if e34402[i]>0 and e384072[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384162[i]>=0
          then othpay2[i]=e384162[i]/e34402[i];
  if e34402[i]>0 and e384072[i]=4 and e384162[i]>=0 then othpay2[i]=e384162[i]/(2*e34402[i]);
  if e34402[i]>0 and e384072[i]=5 and e384162[i]>=0 then othpay2[i]=e384162[i]/(4.3*e34402[i]);
  if e34402[i]>0 and e384072[i]=6 and e384162[i]>=0 and e35600[i]>0 then
          othpay2[i]=e384162[i]/(e35600[i]*e34402[i]);
  if e34402[i]>0 and e384072[i]=8 and e384162[i]>=0 then othpay2[i]=e384162[i]/(2.15*e34402[i]);
   if e384072[i] in (12,13) and e384162[i]>=o then do;
      if e38416i2[i]=3 and e38416ib2[i]>0 and e38416k2[i]>=0 and e34402[i]>0 then
        othpay2[i]=(e384162[i]/e38416ib2[i])*e38416k2[i]/e34402[i];
      if e38416i2[i]=4 and e38416ib2[i]>0 and e34402[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])/e34402[i];
      if e38416i2[i]=5 and e38416ib2[i]>0 and e34402[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])/(e34402[i]*4.3);
    /* missing value */
      if e38416i2[i] in (3,4,5) and -4<e34402[i]<=0 then othpay2[i]=-3;
  end;

  if e384072[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay2[i]=-3;
  if e384072[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          othpay2[i]=e34402[i];

/*for bonuses*/

  if e34402[i]>0 and e384073[i]=1 and e384163[i]>=0 then othpay3[i]=e384163[i];
  if e34402[i]>0 and e384073[i]=2 and e38416k3[i]>0 and e384163[i]>=0 then
          othpay3[i]=(e384163[i]*e38416k3[i])/e34402[i];
  if e34402[i]>0 and e384073[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384163[i]>=0
          then othpay3[i]=e384163[i]/e34402[i];
  if e34402[i]>0 and e384073[i]=4 and e384163[i]>=0 then othpay3[i]=e384163[i]/(2*e34402[i]);
  if e34402[i]>0 and e384073[i]=5 and e384163[i]>=0 then othpay3[i]=e384163[i]/(4.3*e34402[i]);
  if e34402[i]>0 and e384073[i]=6 and e384163[i]>=0 and e35600[i]>0 then
          othpay3[i]=e384163[i]/(e35600[i]*e34402[i]);
  if e34402[i]>0 and e384073[i]=8 and e384163[i]>=0 then othpay3[i]=e384163[i]/(2.15*e34402[i]);
   if e384073[i] in (12,13) and e384163[i]>=o then do;
      if e38416i3[i]=3 and e38416ib3[i]>0 and e38416k3[i]>=0 and e34402[i]>0 then
        othpay3[i]=(e384163[i]/e38416ib3[i])*e38416k3[i]/e34402[i];
      if e38416i3[i]=4 and e38416ib3[i]>0 and e34402[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])/e34402[i];
      if e38416i3[i]=5 and e38416ib3[i]>0 and e34402[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])/(e34402[i]*4.3);
    /* missing value */
      if e38416i3[i] in (3,4,5) and -4<e34402[i]<=0 then othpay3[i]=-3;
  end;

  if e384073[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay3[i]=-3;
```

```
  if e384073[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          othpay3[i]=e34402[i];

 /*for incentive pay*/

  if e34402[i]>0 and e384074[i]=1 and e384164[i]>=0 then othpay4[i]=e384164[i];
  if e34402[i]>0 and e384074[i]=2 and e38416k4[i]>0 and e384164[i]>=0 then
          othpay4[i]=(e384164[i]*e38416k4[i])/e34402[i];
  if e34402[i]>0 and e384074[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384164[i]>=0
          then othpay4[i]=e384164[i]/e34402[i];
  if e34402[i]>0 and e384074[i]=4 and e384164[i]>=0 then othpay4[i]=e384164[i]/(2*e34402[i]);
  if e34402[i]>0 and e384074[i]=5 and e384164[i]>=0 then othpay4[i]=e384164[i]/(4.3*e34402[i]);
  if e34402[i]>0 and e384074[i]=6 and e384164[i]>=0 and e35600[i]>0 then
          othpay4[i]=e384164[i]/(e35600[i]*e34402[i]);
  if e34402[i]>0 and e384074[i]=8 and e384164[i]>=0 then othpay4[i]=e384164[i]/(2.15*e34402[i]);
   if e384074[i] in (12,13) and e384164[i]>=o then do;
      if e38416i4[i]=3 and e38416ib4[i]>0 and e38416k4[i]>=0 and e34402[i]>0 then
        othpay4[i]=(e384164[i]/e38416ib4[i])*e38416k4[i]/e34402[i];
      if e38416i4[i]=4 and e38416ib4[i]>0 and e34402[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])/e34402[i];
      if e38416i4[i]=5 and e38416ib4[i]>0 and e34402[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])/(e34402[i]*4.3);
    /* missing value */
      if e38416i4[i] in (3,4,5) and -4<e34402[i]<=0 then othpay4[i]=-3;
   end;

  if e384074[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay4[i]=-3;
  if e384074[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          othpay4[i]=e34402[i];

 /*for others*/

  if e34402[i]>0 and e384075[i]=1 and e384165[i]>=0 then othpay5[i]=e384165[i];
  if e34402[i]>0 and e384075[i]=2 and e38416k5[i]>0 and e384165[i]>=0 then
          othpay5[i]=(e384165[i]*e38416k5[i])/e34402[i];
  if e34402[i]>0 and e384075[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384165[i]>=0
          then othpay5[i]=e384165[i]/e34402[i];
  if e34402[i]>0 and e384075[i]=4 and e384165[i]>=0 then othpay5[i]=e384165[i]/(2*e34402[i]);
  if e34402[i]>0 and e384075[i]=5 and e384165[i]>=0 then othpay5[i]=e384165[i]/(4.3*e34402[i]);
  if e34402[i]>0 and e384075[i]=6 and e384165[i]>=0 and e35600[i]>0 then
          othpay5[i]=e384165[i]/(e35600[i]*e34402[i]);
  if e34402[i]>0 and e384075[i]=8 and e384165[i]>=0 then othpay5[i]=e384165[i]/(2.15*e34402[i]);
   if e384075[i] in (12,13) and e384165[i]>=o then do;
      if e38416i5[i]=3 and e38416ib5[i]>0 and e38416k5[i]>=0 and e34402[i]>0 then
        othpay5[i]=(e384165[i]/e38416ib5[i])*e38416k5[i]/e34402[i];
      if e38416i5[i]=4 and e38416ib5[i]>0 and e34402[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])/e34402[i];
      if e38416i5[i]=5 and e38416ib5[i]>0 and e34402[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])/(e34402[i]*4.3);
    /* missing value */
      if e38416i5[i] in (3,4,5) and -4<e34402[i]<=0 then othpay5[i]=-3;
   end;

  if e384075[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay5[i]=-3;
  if e384075[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          othpay5[i]=e34402[i];

 end;

/*** case ii. without overtime at the beginning, diff no. of hours.***/
```

```
if e3800b[i]=0 and e212001[i] ne 1 then do;

  /* for tips*/

   if e3800ff[i]>0 and e384071[i]=1 and e384161[i]>=0 then othpay1[i]=e384161[i];
   if e3800ff[i]>0 and e384071[i]=2 and e38416k1[i]>0 and e384161[i]>=0 then
            othpay1[i]=(e384161[i]*e38416k1[i])/e3800ff[i];
   if e3800ff[i]>0 and e384071[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384161[i]>=0
            then othpay1[i]=e384161[i]/e3800ff[i];
   if e3800ff[i]>0 and e384071[i]=4 and e384161[i]>=0 then othpay1[i]=e384161[i]/(2*e3800ff[i]);
   if e3800ff[i]>0 and e384071[i]=5 and e384161[i]>=0 then othpay1[i]=e384161[i]/(4.3*e3800ff[i]);
   if e3800ff[i]>0 and e384071[i]=6 and e384161[i]>=0 and e35600[i]>0 then
            othpay1[i]=e384161[i]/(e35600[i]*e3800ff[i]);
   if e3800ff[i]>0 and e384071[i]=8 and e384161[i]>=0 then othpay1[i]=e384161[i]/(2.15*e3800ff[i]);
   if e384071[i] in (9,14) then othpay1[i]=0;
   if e384071[i] in (12,13) and e384161[i]>=o then do;
      if e38416g1[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
        if e225301[i] in (1,2,4,5) and e225001[i]>0 and othpay1[i]>=0 then
            othpay1[i]=othpay1[i]/e225001[i]*e384161[i];
        if e225301[i]=3 and e225001[i]>0 and e38416k1[i]>0 and e225501[i]>0 and othpay1[i]>=0
           then othpay1[i]=othpay1[i]/(e225001[i]*e225501[i])*e384161[i]*e38416k1[i];
        if e225301[i]=3 and e38416k1[i]<=0 then othpay1[i]=-3;
      end;
      else do; /*in all the other cases */
       if e38416i1[i]=1 and e38416ib1[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])*60;
       if e38416i1[i]=2 and e38416ib1[i]>0 then othpay1[i]=e384161[i]/e38416ib1[i];
       if e38416i1[i]=3 and e38416ib1[i]>0 and e38416k1[i]>=0 and e3800ff[i]>0 then
         othpay1[i]=(e384161[i]/e38416ib1[i])*e38416k1[i]/e3800ff[i];
       if e38416i1[i]=4 and e38416ib1[i]>0 and e3800ff[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])/e3800ff[i];
       if e38416i1[i]=5 and e38416ib1[i]>0 and e3800ff[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])/(e3800ff[i]*4.3);
     /* missing value */
       if -4<e38416i1[i]<0 or -4<e38416ib1[i]<=0 then othpay1[i]=-3;
       if e38416i1[i] in (3,4,5) and -4<e3800ff[i]<=0 then othpay1[i]=-3;
       if e38416i1[i]=3 and e38416k1[i]<0 then othpay1[i]=-3;
      end;
    end;

   if e384071[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384161[i]<0 then
            othpay1[i]=e225001[i];
   if e384071[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e3800ff[i]=0 then othpay1[i]=-3;
   if e384071[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e3800ff[i]<0 then
            othpay1[i]=e3800ff[i];
   if e384071[i]=2 and e38416k1[i] le 0 then othpay1[i]=-3;
   if e384071[i]=2 and -4<e38416k1[i]<0 then othpay1[i]=e225501[i];
   if e384071[i]=6 and e35600[i] le 0 then othpay1[i]=-3;
   if e384071[i]=6 and -4<e35600[i]<0 then othpay1[i]=e35600[i];

 /*for commissions*/

   if e3800ff[i]>0 and e384072[i]=1 and e384162[i]>=0 then othpay2[i]=e384162[i];
   if e3800ff[i]>0 and e384072[i]=2 and e38416k2[i]>0 and e384162[i]>=0 then
            othpay2[i]=(e384162[i]*e38416k2[i])/e3800ff[i];
   if e3800ff[i]>0 and e384072[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384162[i]>=0
            then othpay2[i]=e384162[i]/e3800ff[i];
   if e3800ff[i]>0 and e384072[i]=4 and e384162[i]>=0 then othpay2[i]=e384162[i]/(2*e3800ff[i]);
   if e3800ff[i]>0 and e384072[i]=5 and e384162[i]>=0 then othpay2[i]=e384162[i]/(4.3*e3800ff[i]);
```

```
  if e3800ff[i]>0 and e384072[i]=6 and e384162[i]>=0 and e35600[i]>0 then
          othpay2[i]=e384162[i]/(e35600[i]*e3800ff[i]);
  if e3800ff[i]>0 and e384072[i]=8 and e384162[i]>=0 then othpay2[i]=e384162[i]/(2.15*e3800ff[i]);
  if e384072[i] in (9,14) then othpay2[i]=0;
  if e384072[i] in (12,13) and e384162[i]>=o then do;
     if e38416g2[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
       if e225302[i] in (1,2,4,5) and e225002[i]>0 and othpay2[i]>=0 then
          othpay2[i]=othpay2[i]/e225002[i]*e384162[i];
       if e225302[i]=3 and e225002[i]>0 and e38416k2[i]>0 and e225502[i]>0 and othpay2[i]>=0
         then othpay2[i]=othpay2[i]/(e225002[i]*e225502[i])*e384162[i]*e38416k2[i];
       if e225302[i]=3 and e38416k2[i]<=0 then othpay2[i]=-3;
     end;
     else do; /*in all the other cases */
      if e38416i2[i]=1 and e38416ib2[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])*60;
      if e38416i2[i]=2 and e38416ib2[i]>0 then othpay2[i]=e384162[i]/e38416ib2[i];
      if e38416i2[i]=3 and e38416ib2[i]>0 and e38416k2[i]>=0 and e3800ff[i]>0 then
        othpay2[i]=(e384162[i]/e38416ib2[i])*e38416k2[i]/e3800ff[i];
      if e38416i2[i]=4 and e38416ib2[i]>0 and e3800ff[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])/e3800ff[i];
      if e38416i2[i]=5 and e38416ib2[i]>0 and e3800ff[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])/(e3800ff[i]*4.3);
    /* missing value */
      if -4<e38416i2[i]<0 or -4<e38416ib2[i]<=0 then othpay2[i]=-3;
      if e38416i2[i] in (3,4,5) and -4<e3800ff[i]<=0 then othpay2[i]=-3;
      if e38416i2[i]=3 and e38416k2[i]<0 then othpay2[i]=-3;
     end;
   end;

  if e384072[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384162[i]<0 then
          othpay2[i]=e225002[i];
  if e384072[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e3800ff[i]=0 then othpay2[i]=-3;
  if e384072[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e3800ff[i]<0 then
          othpay2[i]=e3800ff[i];
  if e384072[i]=2 and e38416k2[i] le 0 then othpay2[i]=-3;
  if e384072[i]=2 and -4<e38416k2[i]<0 then othpay2[i]=e225502[i];
  if e384072[i]=6 and e35600[i] le 0 then othpay2[i]=-3;
  if e384072[i]=6 and -4<e35600[i]<0 then othpay2[i]=e35600[i];

/*for bonuses*/

  if e3800ff[i]>0 and e384073[i]=1 and e384163[i]>=0 then othpay3[i]=e384163[i];
  if e3800ff[i]>0 and e384073[i]=2 and e38416k3[i]>0 and e384163[i]>=0 then
          othpay3[i]=(e384163[i]*e38416k3[i])/e3800ff[i];
  if e3800ff[i]>0 and e384073[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384163[i]>=0
          then othpay3[i]=e384163[i]/e3800ff[i];
  if e3800ff[i]>0 and e384073[i]=4 and e384163[i]>=0 then othpay3[i]=e384163[i]/(2*e3800ff[i]);
  if e3800ff[i]>0 and e384073[i]=5 and e384163[i]>=0 then othpay3[i]=e384163[i]/(4.3*e3800ff[i]);
  if e3800ff[i]>0 and e384073[i]=6 and e384163[i]>=0 and e35600[i]>0 then
          othpay3[i]=e384163[i]/(e35600[i]*e3800ff[i]);
  if e3800ff[i]>0 and e384073[i]=8 and e384163[i]>=0 then othpay3[i]=e384163[i]/(2.15*e3800ff[i]);
  if e384073[i] in (9,14) then othpay3[i]=0;
  if e384073[i] in (12,13) and e384163[i]>=o then do;
     if e38416g3[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
       if e225303[i] in (1,2,4,5) and e225003[i]>0 and othpay3[i]>=0 then
          othpay3[i]=othpay3[i]/e225003[i]*e384163[i];
       if e225303[i]=3 and e225003[i]>0 and e38416k3[i]>0 and e225503[i]>0 and othpay3[i]>=0
         then othpay3[i]=othpay3[i]/(e225003[i]*e225503[i])*e384163[i]*e38416k3[i];
       if e225303[i]=3 and e38416k3[i]<=0 then othpay3[i]=-3;
     end;
```

```
    else do; /*in all the other cases */
     if e38416i3[i]=1 and e38416ib3[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])*60;
     if e38416i3[i]=2 and e38416ib3[i]>0 then othpay3[i]=e384163[i]/e38416ib3[i];
     if e38416i3[i]=3 and e38416ib3[i]>0 and e38416k3[i]>=0 and e3800ff[i]>0 then
       othpay3[i]=(e384163[i]/e38416ib3[i])*e38416k3[i]/e3800ff[i];
     if e38416i3[i]=4 and e38416ib3[i]>0 and e3800ff[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])/e3800ff[i];
     if e38416i3[i]=5 and e38416ib3[i]>0 and e3800ff[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])/(e3800ff[i]*4.3);
    /* missing value */
     if -4<e38416i3[i]<0 or -4<e38416ib3[i]<=0 then othpay3[i]=-3;
     if e38416i3[i] in (3,4,5) and -4<e3800ff[i]<=0 then othpay3[i]=-3;
     if e38416i3[i]=3 and e38416k3[i]<0 then othpay3[i]=-3;
    end;
  end;

  if e384073[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384163[i]<0 then
          othpay3[i]=e225003[i];
  if e384073[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e3800ff[i]=0 then othpay3[i]=-3;
  if e384073[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e3800ff[i]<0 then
          othpay3[i]=e3800ff[i];
  if e384073[i]=2 and e38416k3[i] le 0 then othpay3[i]=-3;
  if e384073[i]=2 and -4<e38416k3[i]<0 then othpay3[i]=e225503[i];
  if e384073[i]=6 and e35600[i] le 0 then othpay3[i]=-3;
  if e384073[i]=6 and -4<e35600[i]<0 then othpay3[i]=e35600[i];

/*for incentive pay*/

  if e3800ff[i]>0 and e384074[i]=1 and e384164[i]>=0 then othpay4[i]=e384164[i];
  if e3800ff[i]>0 and e384074[i]=2 and e38416k4[i]>0 and e384164[i]>=0 then
          othpay4[i]=(e384164[i]*e38416k4[i])/e3800ff[i];
  if e3800ff[i]>0 and e384074[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384164[i]>=0
          then othpay4[i]=e384164[i]/e3800ff[i];
  if e3800ff[i]>0 and e384074[i]=4 and e384164[i]>=0 then othpay4[i]=e384164[i]/(2*e3800ff[i]);
  if e3800ff[i]>0 and e384074[i]=5 and e384164[i]>=0 then othpay4[i]=e384164[i]/(4.3*e3800ff[i]);
  if e3800ff[i]>0 and e384074[i]=6 and e384164[i]>=0 and e35600[i]>0 then
          othpay4[i]=e384164[i]/(e35600[i]*e3800ff[i]);
  if e3800ff[i]>0 and e384074[i]=8 and e384164[i]>=0 then othpay4[i]=e384164[i]/(2.15*e3800ff[i]);
  if e384074[i] in (9,14) then othpay4[i]=0;
  if e384074[i] in (12,13) and e384164[i]>=o then do;
    if e38416g4[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
      if e225304[i] in (1,2,4,5) and e225004[i]>0 and othpay4[i]>=0 then
          othpay4[i]=othpay4[i]/e225004[i]*e384164[i];
      if e225304[i]=3 and e225004[i]>0 and e38416k4[i]>0 and e225504[i]>0 and othpay4[i]>=0
        then othpay4[i]=othpay4[i]/(e225004[i]*e225504[i])*e384164[i]*e38416k4[i];
      if e225304[i]=3 and e38416k4[i]<=0 then othpay4[i]=-3;
    end;
    else do; /*in all the other cases */
     if e38416i4[i]=1 and e38416ib4[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])*60;
     if e38416i4[i]=2 and e38416ib4[i]>0 then othpay4[i]=e384164[i]/e38416ib4[i];
     if e38416i4[i]=3 and e38416ib4[i]>0 and e38416k4[i]>=0 and e3800ff[i]>0 then
       othpay4[i]=(e384164[i]/e38416ib4[i])*e38416k4[i]/e3800ff[i];
     if e38416i4[i]=4 and e38416ib4[i]>0 and e3800ff[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])/e3800ff[i];
     if e38416i4[i]=5 and e38416ib4[i]>0 and e3800ff[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])/(e3800ff[i]*4.3);
    /* missing value */
     if -4<e38416i4[i]<0 or -4<e38416ib4[i]<=0 then othpay4[i]=-3;
     if e38416i4[i] in (3,4,5) and -4<e3800ff[i]<=0 then othpay4[i]=-3;
     if e38416i4[i]=3 and e38416k4[i]<0 then othpay4[i]=-3;
    end;
```

```
 end;

 if e384074[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384164[i]<0 then
        othpay4[i]=e225004[i];
 if e384074[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e3800ff[i]=0 then othpay4[i]=-3;
 if e384074[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e3800ff[i]<0 then
        othpay4[i]=e3800ff[i];
 if e384074[i]=2 and e38416k4[i] le 0 then othpay4[i]=-3;
 if e384074[i]=2 and -4<e38416k4[i]<0 then othpay4[i]=e225504[i];
 if e384074[i]=6 and e35600[i] le 0 then othpay4[i]=-3;
 if e384074[i]=6 and -4<e35600[i]<0 then othpay4[i]=e35600[i];

/*for others*/

 if e3800ff[i]>0 and e384075[i]=1 and e384165[i]>=0 then othpay5[i]=e384165[i];
 if e3800ff[i]>0 and e384075[i]=2 and e38416k5[i]>0 and e384165[i]>=0 then
        othpay5[i]=(e384165[i]*e38416k5[i])/e3800ff[i];
 if e3800ff[i]>0 and e384075[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384165[i]>=0
        then othpay5[i]=e384165[i]/e3800ff[i];
 if e3800ff[i]>0 and e384075[i]=4 and e384165[i]>=0 then othpay5[i]=e384165[i]/(2*e3800ff[i]);
 if e3800ff[i]>0 and e384075[i]=5 and e384165[i]>=0 then othpay5[i]=e384165[i]/(4.3*e3800ff[i]);
 if e3800ff[i]>0 and e384075[i]=6 and e384165[i]>=0 and e35600[i]>0 then
        othpay5[i]=e384165[i]/(e35600[i]*e3800ff[i]);
 if e3800ff[i]>0 and e384075[i]=8 and e384165[i]>=0 then othpay5[i]=e384165[i]/(2.15*e3800ff[i]);
 if e384075[i] in (9,14) then othpay5[i]=0;
 if e384075[i] in (12,13) and e384165[i]>=o then do;
    if e38416g5[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
      if e225305[i] in (1,2,4,5) and e225005[i]>0 and othpay5[i]>=0 then
          othpay5[i]=othpay5[i]/e225005[i]*e384165[i];
      if e225305[i]=3 and e225005[i]>0 and e38416k5[i]>0 and e225505[i]>0 and othpay5[i]>=0
         then othpay5[i]=othpay5[i]/(e225005[i]*e225505[i])*e384165[i]*e38416k5[i];
      if e225305[i]=3 and e38416k5[i]<=0 then othpay5[i]=-3;
    end;
    else do; /*in all the other cases */
     if e38416i5[i]=1 and e38416ib5[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])*60;
     if e38416i5[i]=2 and e38416ib5[i]>0 then othpay5[i]=e384165[i]/e38416ib5[i];
     if e38416i5[i]=3 and e38416ib5[i]>0 and e38416k5[i]>=0 and e3800ff[i]>0 then
       othpay5[i]=(e384165[i]/e38416ib5[i])*e38416k5[i]/e3800ff[i];
     if e38416i5[i]=4 and e38416ib5[i]>0 and e3800ff[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])/e3800ff[i];
     if e38416i5[i]=5 and e38416ib5[i]>0 and e3800ff[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])/(e3800ff[i]*4.3);
   /* missing value */
     if -4<e38416i5[i]<0 or -4<e38416ib5[i]<=0 then othpay5[i]=-3;
     if e38416i5[i] in (3,4,5) and -4<e3800ff[i]<=0 then othpay5[i]=-3;
     if e38416i5[i]=3 and e38416k5[i]<0 then othpay5[i]=-3;
    end;
 end;

 if e384075[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384165[i]<0 then
        othpay5[i]=e225005[i];
 if e384075[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e3800ff[i]=0 then othpay5[i]=-3;
 if e384075[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e3800ff[i]<0 then
        othpay5[i]=e3800ff[i];
 if e384075[i]=2 and e38416k5[i] le 0 then othpay5[i]=-3;
 if e384075[i]=2 and -4<e38416k5[i]<0 then othpay5[i]=e225505[i];
 if e384075[i]=6 and e35600[i] le 0 then othpay5[i]=-3;
 if e384075[i]=6 and -4<e35600[i]<0 then othpay5[i]=e35600[i];
```

```
end;

/*** case iii. only one compensation at the beginning,same no. of hours. ***/

if e38329[i]=1 and e20700[i]=1 then do;

  /* for tips*/

  if e212002[i]=1 then do;

    /*with overtime at the beginning*/
    if e38102[i] ne 1 and e38102[i] ne 3 then do;

    /*report hourly wage at the beginning*/
    if e23901[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay1[i]=e38329d[i];
    if e23901[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
            othpay1[i]=(e38329d[i]*e38329k[i])/e23901[i];
    if e23901[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
            then othpay1[i]=e38329d[i]/e23901[i];
    if e23901[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(2*e23901[i]);
    if e23901[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(4.3*e23901[i]);
    if e23901[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
            othpay1[i]=e38329d[i]/(e35600[i]*e23901[i]);
    if e23901[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(2.15*e23901[i]);
    if e38329b[i] in (9,14) then othpay1[i]=0;
    if e38329b[i] in (12,13) and e38329d[i]>=o then do;
       if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
         if e225301[i] in (1,2,4,5) and e225001[i]>0 and othpay1[i]>=0 then
             othpay1[i]=othpay1[i]/e225001[i]*e38329d[i];
         if e225301[i]=3 and e225001[i]>0 and e38329k[i]>0 and e225501[i]>0 and othpay1[i]>=0
            then othpay1[i]=othpay1[i]/(e225001[i]*e225501[i])*e38329d[i]*e38329k[i];
         if e225301[i]=3 and e38329k[i]<=0 then othpay1[i]=-3;
       end;
       else do; /*in all the other cases */
        if e38329i[i]=1 and e38329ib[i]>0 then othpay1[i]=(e38329d[i]/e38329ib[i])*60;
        if e38329i[i]=2 and e38329ib[i]>0 then othpay1[i]=e38329d[i]/e38329ib[i];
        if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e23901[i]>0 then
          othpay1[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e23901[i];
        if e38329i[i]=4 and e38329ib[i]>0 and e23901[i]>0 then othpay1[i]=(e38329d[i]/e38329ib[i])/e23901[i];
        if e38329i[i]=5 and e38329ib[i]>0 and e23901[i]>0 then othpay1[i]=(e38329d[i]/e38329ib[i])/(e23901[i]*4.3);
      /* missing value */
        if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay1[i]=-3;
        if e38329i[i] in (3,4,5) and -4<e23901[i]<=0 then othpay1[i]=-3;
        if e38329i[i]=3 and e38329k[i]<0 then othpay1[i]=-3;
       end;
    end;

    if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
            othpay1[i]=e225001[i];
    if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay1[i]=-3;
    if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
            othpay1[i]=e23901[i];
    if e38329b[i]=2 and e38329k[i] le 0 then othpay1[i]=-3;
    if e38329b[i]=2 and -4<e38329k[i]<0 then othpay1[i]=e225501[i];
    if e38329b[i]=6 and e35600[i] le 0 then othpay1[i]=-3;
    if e38329b[i]=6 and -4<e35600[i]<0 then othpay1[i]=e35600[i];
```

```
/*report nonhourly wage*/
if e34428[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay1[i]=e38329d[i];
if e34428[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
        othpay1[i]=(e38329d[i]*e38329k[i])/e34428[i];
if e34428[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
        then othpay1[i]=e38329d[i]/e34428[i];
if e34428[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(2*e34428[i]);
if e34428[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(4.3*e34428[i]);
if e34428[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
        othpay1[i]=e38329d[i]/(e35600[i]*e34428[i]);
if e34428[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(2.15*e34428[i]);
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
    if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e34428[i]>0 then
      othpay1[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e34428[i];
    if e38329i[i]=4 and e38329ib[i]>0 and e34428[i]>0 then othpay1[i]=(e38329d[i]/e38329ib[i])/e34428[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e34428[i]>0 then othpay1[i]=(e38329d[i]/e38329ib[i])/(e34428[i]*4.3);
  /* missing value */
    if e38329i[i] in (3,4,5) and -4<e34428[i]<=0 then othpay1[i]=-3;
 end;

if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay1[i]=-3;
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay1[i]=e34428[i];

end;

/*without overtime at the beginning*/
if e3800b[i]=1 then do;

/*report hourly wage at the beginning*/
if e23901[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay1[i]=e38329d[i];
if e23901[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
        othpay1[i]=(e38329d[i]*e38329k[i])/e23901[i];
if e23901[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
        then othpay1[i]=e38329d[i]/e23901[i];
if e23901[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(2*e23901[i]);
if e23901[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(4.3*e23901[i]);
if e23901[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
        othpay1[i]=e38329d[i]/(e35600[i]*e23901[i]);
if e23901[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(2.15*e23901[i]);
if e38329b[i] in (9,14) then othpay1[i]=0;
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
   if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
      if e225301[i] in (1,2,4,5) and e225001[i]>0 and othpay1[i]>=0 then
          othpay1[i]=othpay1[i]/e225001[i]*e38329d[i];
      if e225301[i]=3 and e225001[i]>0 and e38329k[i]>0 and e225501[i]>0 and othpay1[i]>=0
         then othpay1[i]=othpay1[i]/(e225001[i]*e225501[i])*e38329d[i]*e38329k[i];
      if e225301[i]=3 and e38329k[i]<=0 then othpay1[i]=-3;
   end;
   else do; /*in all the other cases */
    if e38329i[i]=1 and e38329ib[i]>0 then othpay1[i]=(e38329d[i]/e38329ib[i])*60;
    if e38329i[i]=2 and e38329ib[i]>0 then othpay1[i]=e38329d[i]/e38329ib[i];
    if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e23901[i]>0 then
      othpay1[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e23901[i];
    if e38329i[i]=4 and e38329ib[i]>0 and e23901[i]>0 then othpay1[i]=(e38329d[i]/e38329ib[i])/e23901[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e23901[i]>0 then othpay1[i]=(e38329d[i]/e38329ib[i])/(e23901[i]*4.3);
```

```
    /* missing value */
      if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay1[i]=-3;
      if e38329i[i] in (3,4,5) and -4<e23901[i]<=0 then othpay1[i]=-3;
      if e38329i[i]=3 and e38329k[i]<0 then othpay1[i]=-3;
     end;
  end;

  if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
          othpay1[i]=e225001[i];
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay1[i]=-3;
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
          othpay1[i]=e23901[i];
  if e38329b[i]=2 and e38329k[i] le 0 then othpay1[i]=-3;
  if e38329b[i]=2 and -4<e38329k[i]<0 then othpay1[i]=e225501[i];
  if e38329b[i]=6 and e35600[i] le 0 then othpay1[i]=-3;
  if e38329b[i]=6 and -4<e35600[i]<0 then othpay1[i]=e35600[i];

  /*report non-hourly wage at the beginning*/
  if e34402[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay1[i]=e38329d[i];
  if e34402[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
          othpay1[i]=(e38329d[i]*e38329k[i])/e34402[i];
  if e34402[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
          then othpay1[i]=e38329d[i]/e34402[i];
  if e34402[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(2*e34402[i]);
  if e34402[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(4.3*e34402[i]);
  if e34402[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
          othpay1[i]=e38329d[i]/(e35600[i]*e34402[i]);
  if e34402[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(2.15*e34402[i]);
  if e38329b[i] in (12,13) and e38329d[i]>=o then do;
      if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e34402[i]>0 then
        othpay1[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e34402[i];
      if e38329i[i]=4 and e38329ib[i]>0 and e34402[i]>0 then othpay1[i]=(e38329d[i]/e38329ib[i])/e34402[i];
      if e38329i[i]=5 and e38329ib[i]>0 and e34402[i]>0 then othpay1[i]=(e38329d[i]/e38329ib[i])/(e34402[i]*4.3);
    /* missing value */
      if e38329i[i] in (3,4,5) and -4<e34402[i]<=0 then othpay1[i]=-3;
  end;

  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay1[i]=-3;
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          othpay1[i]=e34402[i];

  end;

 end;

/*for commissions*/

 if e212003[i]=1 then do;

  /*with overtime at the beginning*/
  if e38102[i] ne 1 and e38102[i] ne 3 then do;

  /*report hourly wage at the beginning*/
  if e23901[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay2[i]=e38329d[i];
  if e23901[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
          othpay2[i]=(e38329d[i]*e38329k[i])/e23901[i];
```

```
if e23901[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
        then othpay2[i]=e38329d[i]/e23901[i];
if e23901[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(2*e23901[i]);
if e23901[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(4.3*e23901[i]);
if e23901[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
        othpay2[i]=e38329d[i]/(e35600[i]*e23901[i]);
if e23901[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(2.15*e23901[i]);
if e38329b[i] in (9,14) then othpay2[i]=0;
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
   if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
     if e225302[i] in (1,2,4,5) and e225002[i]>0 and othpay2[i]>=0 then
        othpay2[i]=othpay2[i]/e225002[i]*e38329d[i];
     if e225302[i]=3 and e225002[i]>0 and e38329k[i]>0 and e225502[i]>0 and othpay2[i]>=0
       then othpay2[i]=othpay2[i]/(e225002[i]*e225502[i])*e38329d[i]*e38329k[i];
     if e225302[i]=3 and e38329k[i]<=0 then othpay2[i]=-3;
   end;
   else do; /*in all the other cases */
    if e38329i[i]=1 and e38329ib[i]>0 then othpay2[i]=(e38329d[i]/e38329ib[i])*60;
    if e38329i[i]=2 and e38329ib[i]>0 then othpay2[i]=e38329d[i]/e38329ib[i];
    if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e23901[i]>0 then
      othpay2[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e23901[i];
    if e38329i[i]=4 and e38329ib[i]>0 and e23901[i]>0 then othpay2[i]=(e38329d[i]/e38329ib[i])/e23901[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e23901[i]>0 then othpay2[i]=(e38329d[i]/e38329ib[i])/(e23901[i]*4.3);
  /* missing value */
    if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay2[i]=-3;
    if e38329i[i] in (3,4,5) and -4<e23901[i]<=0 then othpay2[i]=-3;
    if e38329i[i]=3 and e38329k[i]<0 then othpay2[i]=-3;
   end;
 end;

if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
        othpay2[i]=e225002[i];
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay2[i]=-3;
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay2[i]=e23901[i];
if e38329b[i]=2 and e38329k[i] le 0 then othpay2[i]=-3;
if e38329b[i]=2 and -4<e38329k[i]<0 then othpay2[i]=e225502[i];
if e38329b[i]=6 and e35600[i] le 0 then othpay2[i]=-3;
if e38329b[i]=6 and -4<e35600[i]<0 then othpay2[i]=e35600[i];

/*report nonhourly wage*/
if e34428[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay2[i]=e38329d[i];
if e34428[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
        othpay2[i]=(e38329d[i]*e38329k[i])/e34428[i];
if e34428[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
        then othpay2[i]=e38329d[i]/e34428[i];
if e34428[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(2*e34428[i]);
if e34428[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(4.3*e34428[i]);
if e34428[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
        othpay2[i]=e38329d[i]/(e35600[i]*e34428[i]);
if e34428[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(2.15*e34428[i]);
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
    if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e34428[i]>0 then
      othpay2[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e34428[i];
    if e38329i[i]=4 and e38329ib[i]>0 and e34428[i]>0 then othpay2[i]=(e38329d[i]/e38329ib[i])/e34428[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e34428[i]>0 then othpay2[i]=(e38329d[i]/e38329ib[i])/(e34428[i]*4.3);
  /* missing value */
```

```
    if e38329i[i] in (3,4,5) and -4<e34428[i]<=0 then othpay2[i]=-3;
 end;

if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay2[i]=-3;
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay2[i]=e34428[i];

end;

/*without overtime at the beginning*/
if e3800b[i]=1 then do;

/*report hourly wage at the beginning*/
if e23901[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay2[i]=e38329d[i];
if e23901[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
        othpay2[i]=(e38329d[i]*e38329k[i])/e23901[i];
if e23901[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
        then othpay2[i]=e38329d[i]/e23901[i];
if e23901[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(2*e23901[i]);
if e23901[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(4.3*e23901[i]);
if e23901[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
        othpay2[i]=e38329d[i]/(e35600[i]*e23901[i]);
if e23901[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(2.15*e23901[i]);
if e38329b[i] in (9,14) then othpay2[i]=0;
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
   if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
     if e225301[i] in (1,2,4,5) and e225001[i]>0 and othpay2[i]>=0 then
        othpay2[i]=othpay2[i]/e225001[i]*e38329d[i];
     if e225301[i]=3 and e225001[i]>0 and e38329k[i]>0 and e225501[i]>0 and othpay2[i]>=0
       then othpay2[i]=othpay2[i]/(e225001[i]*e225501[i])*e38329d[i]*e38329k[i];
     if e225301[i]=3 and e38329k[i]<=0 then othpay2[i]=-3;
   end;
   else do; /*in all the other cases */
    if e38329i[i]=1 and e38329ib[i]>0 then othpay2[i]=(e38329d[i]/e38329ib[i])*60;
    if e38329i[i]=2 and e38329ib[i]>0 then othpay2[i]=e38329d[i]/e38329ib[i];
    if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e23901[i]>0 then
      othpay2[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e23901[i];
    if e38329i[i]=4 and e38329ib[i]>0 and e23901[i]>0 then othpay2[i]=(e38329d[i]/e38329ib[i])/e23901[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e23901[i]>0 then othpay2[i]=(e38329d[i]/e38329ib[i])/(e23901[i]*4.3);
  /* missing value */
    if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay2[i]=-3;
    if e38329i[i] in (3,4,5) and -4<e23901[i]<=0 then othpay2[i]=-3;
    if e38329i[i]=3 and e38329k[i]<0 then othpay2[i]=-3;
   end;
 end;

if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
        othpay2[i]=e225001[i];
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay2[i]=-3;
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay2[i]=e23901[i];
if e38329b[i]=2 and e38329k[i] le 0 then othpay2[i]=-3;
if e38329b[i]=2 and -4<e38329k[i]<0 then othpay2[i]=e225501[i];
if e38329b[i]=6 and e35600[i] le 0 then othpay2[i]=-3;
if e38329b[i]=6 and -4<e35600[i]<0 then othpay2[i]=e35600[i];

/*report nonhourly wage*/
```

```
 if e34402[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay2[i]=e38329d[i];
 if e34402[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
          othpay2[i]=(e38329d[i]*e38329k[i])/e34402[i];
 if e34402[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
          then othpay2[i]=e38329d[i]/e34402[i];
 if e34402[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(2*e34402[i]);
 if e34402[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(4.3*e34402[i]);
 if e34402[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
          othpay2[i]=e38329d[i]/(e35600[i]*e34402[i]);
 if e34402[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(2.15*e34402[i]);
 if e38329b[i] in (12,13) and e38329d[i]>=o then do;
     if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e34402[i]>0 then
       othpay2[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e34402[i];
     if e38329i[i]=4 and e38329ib[i]>0 and e34402[i]>0 then othpay2[i]=(e38329d[i]/e38329ib[i])/e34402[i];
     if e38329i[i]=5 and e38329ib[i]>0 and e34402[i]>0 then othpay2[i]=(e38329d[i]/e38329ib[i])/(e34402[i]*4.3);
   /* missing value */
     if e38329i[i] in (3,4,5) and -4<e34402[i]<=0 then othpay2[i]=-3;
  end;

 if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay2[i]=-3;
 if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          othpay2[i]=e34402[i];

 end;
end;

/*for bonuses*/

if e212004[i]=1 then do;

 /*with overtime at the beginning*/
 if e38102[i] ne 1 and e38102[i] ne 3 then do;

 /* report hourly wage at the beginning*/
 if e23901[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay3[i]=e38329d[i];
 if e23901[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
          othpay3[i]=(e38329d[i]*e38329k[i])/e23901[i];
 if e23901[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
          then othpay3[i]=e38329d[i]/e23901[i];
 if e23901[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(2*e23901[i]);
 if e23901[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(4.3*e23901[i]);
 if e23901[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
          othpay3[i]=e38329d[i]/(e35600[i]*e23901[i]);
 if e23901[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(2.15*e23901[i]);
 if e38329b[i] in (9,14) then othpay3[i]=0;
 if e38329b[i] in (12,13) and e38329d[i]>=o then do;
    if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
      if e225303[i] in (1,2,4,5) and e225003[i]>0 and othpay3[i]>=0 then
          othpay3[i]=othpay3[i]/e225003[i]*e38329d[i];
      if e225303[i]=3 and e225003[i]>0 and e38329k[i]>0 and e225503[i]>0 and othpay3[i]>=0
         then othpay3[i]=othpay3[i]/(e225003[i]*e225503[i])*e38329d[i]*e38329k[i];
      if e225303[i]=3 and e38329k[i]<=0 then othpay3[i]=-3;
    end;
    else do; /*in all the other cases */
     if e38329i[i]=1 and e38329ib[i]>0 then othpay3[i]=(e38329d[i]/e38329ib[i])*60;
     if e38329i[i]=2 and e38329ib[i]>0 then othpay3[i]=e38329d[i]/e38329ib[i];
     if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e23901[i]>0 then
```

```
      othpay3[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e23901[i];
    if e38329i[i]=4 and e38329ib[i]>0 and e23901[i]>0 then othpay3[i]=(e38329d[i]/e38329ib[i])/e23901[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e23901[i]>0 then othpay3[i]=(e38329d[i]/e38329ib[i])/(e23901[i]*4.3);
  /* missing value */
    if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay3[i]=-3;
    if e38329i[i] in (3,4,5) and -4<e23901[i]<=0 then othpay3[i]=-3;
    if e38329i[i]=3 and e38329k[i]<0 then othpay3[i]=-3;
   end;
 end;

if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
        othpay3[i]=e225003[i];
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay3[i]=-3;
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay3[i]=e23901[i];
if e38329b[i]=2 and e38329k[i] le 0 then othpay3[i]=-3;
if e38329b[i]=2 and -4<e38329k[i]<0 then othpay3[i]=e225503[i];
if e38329b[i]=6 and e35600[i] le 0 then othpay3[i]=-3;
if e38329b[i]=6 and -4<e35600[i]<0 then othpay3[i]=e35600[i];

/*report non-hourly wage*/
if e34428[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay3[i]=e38329d[i];
if e34428[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
        othpay3[i]=(e38329d[i]*e38329k[i])/e34428[i];
if e34428[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
        then othpay3[i]=e38329d[i]/e34428[i];
if e34428[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(2*e34428[i]);
if e34428[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(4.3*e34428[i]);
if e34428[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
        othpay3[i]=e38329d[i]/(e35600[i]*e34428[i]);
if e34428[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(2.15*e34428[i]);
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
    if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e34428[i]>0 then
      othpay3[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e34428[i];
    if e38329i[i]=4 and e38329ib[i]>0 and e34428[i]>0 then othpay3[i]=(e38329d[i]/e38329ib[i])/e34428[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e34428[i]>0 then othpay3[i]=(e38329d[i]/e38329ib[i])/(e34428[i]*4.3);
  /* missing value */
    if e38329i[i] in (3,4,5) and -4<e34428[i]<=0 then othpay3[i]=-3;
 end;

if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay3[i]=-3;
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay3[i]=e34428[i];

end;

/*without overtime at the beginning*/
if e3800b[i]=1 then do;

/* report hourly wage at the beginning*/
if e23901[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay3[i]=e38329d[i];
if e23901[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
        othpay3[i]=(e38329d[i]*e38329k[i])/e23901[i];
if e23901[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
        then othpay3[i]=e38329d[i]/e23901[i];
if e23901[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(2*e23901[i]);
if e23901[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(4.3*e23901[i]);
```

```
if e23901[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
        othpay3[i]=e38329d[i]/(e35600[i]*e23901[i]);
if e23901[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(2.15*e23901[i]);
if e38329b[i] in (9,14) then othpay3[i]=0;
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
   if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
     if e225301[i] in (1,2,4,5) and e225001[i]>0 and othpay3[i]>=0 then
        othpay3[i]=othpay3[i]/e225001[i]*e38329d[i];
     if e225301[i]=3 and e225001[i]>0 and e38329k[i]>0 and e225501[i]>0 and othpay3[i]>=0
       then othpay3[i]=othpay3[i]/(e225001[i]*e225501[i])*e38329d[i]*e38329k[i];
     if e225301[i]=3 and e38329k[i]<=0 then othpay3[i]=-3;
   end;
   else do; /*in all the other cases */
    if e38329i[i]=1 and e38329ib[i]>0 then othpay3[i]=(e38329d[i]/e38329ib[i])*60;
    if e38329i[i]=2 and e38329ib[i]>0 then othpay3[i]=e38329d[i]/e38329ib[i];
    if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e23901[i]>0 then
      othpay3[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e23901[i];
    if e38329i[i]=4 and e38329ib[i]>0 and e23901[i]>0 then othpay3[i]=(e38329d[i]/e38329ib[i])/e23901[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e23901[i]>0 then othpay3[i]=(e38329d[i]/e38329ib[i])/(e23901[i]*4.3);
  /* missing value */
    if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay3[i]=-3;
    if e38329i[i] in (3,4,5) and -4<e23901[i]<=0 then othpay3[i]=-3;
    if e38329i[i]=3 and e38329k[i]<0 then othpay3[i]=-3;
   end;
 end;

if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
        othpay3[i]=e225001[i];
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay3[i]=-3;
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay3[i]=e23901[i];
if e38329b[i]=2 and e38329k[i] le 0 then othpay3[i]=-3;
if e38329b[i]=2 and -4<e38329k[i]<0 then othpay3[i]=e225501[i];
if e38329b[i]=6 and e35600[i] le 0 then othpay3[i]=-3;
if e38329b[i]=6 and -4<e35600[i]<0 then othpay3[i]=e35600[i];

/*report non-hourly wage*/
if e34402[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay3[i]=e38329d[i];
if e34402[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
        othpay3[i]=(e38329d[i]*e38329k[i])/e34402[i];
if e34402[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
        then othpay3[i]=e38329d[i]/e34402[i];
if e34402[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(2*e34402[i]);
if e34402[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(4.3*e34402[i]);
if e34402[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
        othpay3[i]=e38329d[i]/(e35600[i]*e34402[i]);
if e34402[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(2.15*e34402[i]);
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
    if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e34402[i]>0 then
      othpay3[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e34402[i];
    if e38329i[i]=4 and e38329ib[i]>0 and e34402[i]>0 then othpay3[i]=(e38329d[i]/e38329ib[i])/e34402[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e34402[i]>0 then othpay3[i]=(e38329d[i]/e38329ib[i])/(e34402[i]*4.3);
  /* missing value */
    if e38329i[i] in (3,4,5) and -4<e34402[i]<=0 then othpay3[i]=-3;
 end;

if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay3[i]=-3;
```

```
 if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          othpay3[i]=e34402[i];

 end;
end;

/*for incentive pay*/

 if e212005[i]=1 then do;

  /*with overtime at the beginning*/
  if e38102[i] ne 1 and e38102[i] ne 3 then do;

  /* report hourly wage at the beginning*/
  if e23901[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay4[i]=e38329d[i];
  if e23901[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
          othpay4[i]=(e38329d[i]*e38329k[i])/e23901[i];
  if e23901[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
          then othpay4[i]=e38329d[i]/e23901[i];
  if e23901[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(2*e23901[i]);
  if e23901[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(4.3*e23901[i]);
  if e23901[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
          othpay4[i]=e38329d[i]/(e35600[i]*e23901[i]);
  if e23901[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(2.15*e23901[i]);
  if e38329b[i] in (9,14) then othpay4[i]=0;
  if e38329b[i] in (12,13) and e38329d[i]>=o then do;
     if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
       if e225304[i] in (1,2,4,5) and e225004[i]>0 and othpay4[i]>=0 then
          othpay4[i]=othpay4[i]/e225004[i]*e38329d[i];
       if e225304[i]=3 and e225004[i]>0 and e38329k[i]>0 and e225504[i]>0 and othpay4[i]>=0
         then othpay4[i]=othpay4[i]/(e225004[i]*e225504[i])*e38329d[i]*e38329k[i];
       if e225304[i]=3 and e38329k[i]<=0 then othpay4[i]=-3;
     end;
     else do; /*in all the other cases */
      if e38329i[i]=1 and e38329ib[i]>0 then othpay4[i]=(e38329d[i]/e38329ib[i])*60;
      if e38329i[i]=2 and e38329ib[i]>0 then othpay4[i]=e38329d[i]/e38329ib[i];
      if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e23901[i]>0 then
        othpay4[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e23901[i];
      if e38329i[i]=4 and e38329ib[i]>0 and e23901[i]>0 then othpay4[i]=(e38329d[i]/e38329ib[i])/e23901[i];
      if e38329i[i]=5 and e38329ib[i]>0 and e23901[i]>0 then othpay4[i]=(e38329d[i]/e38329ib[i])/(e23901[i]*4.3);
    /* missing value */
      if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay4[i]=-3;
      if e38329i[i] in (3,4,5) and -4<e23901[i]<=0 then othpay4[i]=-3;
      if e38329i[i]=3 and e38329k[i]<0 then othpay4[i]=-3;
     end;
   end;

  if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
          othpay4[i]=e225004[i];
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay4[i]=-3;
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
          othpay4[i]=e23901[i];
  if e38329b[i]=2 and e38329k[i] le 0 then othpay4[i]=-3;
  if e38329b[i]=2 and -4<e38329k[i]<0 then othpay4[i]=e225504[i];
  if e38329b[i]=6 and e35600[i] le 0 then othpay4[i]=-3;
  if e38329b[i]=6 and -4<e35600[i]<0 then othpay4[i]=e35600[i];
```

```
/* report nonhourly wage*/
if e34428[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay4[i]=e38329d[i];
if e34428[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
        othpay4[i]=(e38329d[i]*e38329k[i])/e34428[i];
if e34428[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
        then othpay4[i]=e38329d[i]/e34428[i];
if e34428[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(2*e34428[i]);
if e34428[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(4.3*e34428[i]);
if e34428[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
        othpay4[i]=e38329d[i]/(e35600[i]*e34428[i]);
if e34428[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(2.15*e34428[i]);
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
    if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e34428[i]>0 then
      othpay4[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e34428[i];
    if e38329i[i]=4 and e38329ib[i]>0 and e34428[i]>0 then othpay4[i]=(e38329d[i]/e38329ib[i])/e34428[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e34428[i]>0 then othpay4[i]=(e38329d[i]/e38329ib[i])/(e34428[i]*4.3);
  /* missing value */
    if e38329i[i] in (3,4,5) and -4<e34428[i]<=0 then othpay4[i]=-3;
 end;

if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay4[i]=-3;
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay4[i]=e34428[i];

end;

/*without overtime at the beginning*/
if e3800b[i]=1 then do;

/* report hourly wage at the beginning*/
if e23901[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay4[i]=e38329d[i];
if e23901[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
        othpay4[i]=(e38329d[i]*e38329k[i])/e23901[i];
if e23901[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
        then othpay4[i]=e38329d[i]/e23901[i];
if e23901[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(2*e23901[i]);
if e23901[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(4.3*e23901[i]);
if e23901[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
        othpay4[i]=e38329d[i]/(e35600[i]*e23901[i]);
if e23901[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(2.15*e23901[i]);
if e38329b[i] in (9,14) then othpay4[i]=0;
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
   if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
      if e225301[i] in (1,2,4,5) and e225001[i]>0 and othpay4[i]>=0 then
          othpay4[i]=othpay4[i]/e225001[i]*e38329d[i];
      if e225301[i]=3 and e225001[i]>0 and e38329k[i]>0 and e225501[i]>0 and othpay4[i]>=0
         then othpay4[i]=othpay4[i]/(e225001[i]*e225501[i])*e38329d[i]*e38329k[i];
      if e225301[i]=3 and e38329k[i]<=0 then othpay4[i]=-3;
   end;
   else do; /*in all the other cases */
    if e38329i[i]=1 and e38329ib[i]>0 then othpay4[i]=(e38329d[i]/e38329ib[i])*60;
    if e38329i[i]=2 and e38329ib[i]>0 then othpay4[i]=e38329d[i]/e38329ib[i];
    if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e23901[i]>0 then
      othpay4[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e23901[i];
    if e38329i[i]=4 and e38329ib[i]>0 and e23901[i]>0 then othpay4[i]=(e38329d[i]/e38329ib[i])/e23901[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e23901[i]>0 then othpay4[i]=(e38329d[i]/e38329ib[i])/(e23901[i]*4.3);
  /* missing value */
```

```
      if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay4[i]=-3;
      if e38329i[i] in (3,4,5) and -4<e23901[i]<=0 then othpay4[i]=-3;
      if e38329i[i]=3 and e38329k[i]<0 then othpay4[i]=-3;
     end;
   end;

  if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
          othpay4[i]=e225001[i];
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay4[i]=-3;
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
          othpay4[i]=e23901[i];
  if e38329b[i]=2 and e38329k[i] le 0 then othpay4[i]=-3;
  if e38329b[i]=2 and -4<e38329k[i]<0 then othpay4[i]=e225501[i];
  if e38329b[i]=6 and e35600[i] le 0 then othpay4[i]=-3;
  if e38329b[i]=6 and -4<e35600[i]<0 then othpay4[i]=e35600[i];

  /* report nonhourly wage*/
  if e34402[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay4[i]=e38329d[i];
  if e34402[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
          othpay4[i]=(e38329d[i]*e38329k[i])/e34402[i];
  if e34402[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
          then othpay4[i]=e38329d[i]/e34402[i];
  if e34402[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(2*e34402[i]);
  if e34402[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(4.3*e34402[i]);
  if e34402[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
          othpay4[i]=e38329d[i]/(e35600[i]*e34402[i]);
  if e34402[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(2.15*e34402[i]);
  if e38329b[i] in (12,13) and e38329d[i]>=o then do;
      if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e34402[i]>0 then
        othpay4[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e34402[i];
      if e38329i[i]=4 and e38329ib[i]>0 and e34402[i]>0 then othpay4[i]=(e38329d[i]/e38329ib[i])/e34402[i];
      if e38329i[i]=5 and e38329ib[i]>0 and e34402[i]>0 then othpay4[i]=(e38329d[i]/e38329ib[i])/(e34402[i]*4.3);
    /* missing value */
      if e38329i[i] in (3,4,5) and -4<e34402[i]<=0 then othpay4[i]=-3;
   end;

  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay4[i]=-3;
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          othpay4[i]=e34402[i];

  end;
 end;

/*for others*/

 if e212006[i]=1 then do;

  /*with overtime at the beginning*/
  if e38102[i] ne 1 and e38102[i] ne 3 then do;

  /* report hourly wage at the beginning*/
  if e23901[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay5[i]=e38329d[i];
  if e23901[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
          othpay5[i]=(e38329d[i]*e38329k[i])/e23901[i];
  if e23901[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
          then othpay5[i]=e38329d[i]/e23901[i];
  if e23901[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(2*e23901[i]);
```

```
if e23901[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(4.3*e23901[i]);
if e23901[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
        othpay5[i]=e38329d[i]/(e35600[i]*e23901[i]);
if e23901[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(2.15*e23901[i]);
if e38329b[i] in (9,14) then othpay5[i]=0;
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
   if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
     if e225305[i] in (1,2,4,5) and e225005[i]>0 and othpay5[i]>=0 then
        othpay5[i]=othpay5[i]/e225005[i]*e38329d[i];
     if e225305[i]=3 and e225005[i]>0 and e38329k[i]>0 and e225505[i]>0 and othpay5[i]>=0
       then othpay5[i]=othpay5[i]/(e225005[i]*e225505[i])*e38329d[i]*e38329k[i];
     if e225305[i]=3 and e38329k[i]<=0 then othpay5[i]=-3;
   end;
   else do; /*in all the other cases */
    if e38329i[i]=1 and e38329ib[i]>0 then othpay5[i]=(e38329d[i]/e38329ib[i])*60;
    if e38329i[i]=2 and e38329ib[i]>0 then othpay5[i]=e38329d[i]/e38329ib[i];
    if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e23901[i]>0 then
      othpay5[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e23901[i];
    if e38329i[i]=4 and e38329ib[i]>0 and e23901[i]>0 then othpay5[i]=(e38329d[i]/e38329ib[i])/e23901[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e23901[i]>0 then othpay5[i]=(e38329d[i]/e38329ib[i])/(e23901[i]*4.3);
  /* missing value */
    if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay5[i]=-3;
    if e38329i[i] in (3,4,5) and -4<e23901[i]<=0 then othpay5[i]=-3;
    if e38329i[i]=3 and e38329k[i]<0 then othpay5[i]=-3;
   end;
 end;

if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
        othpay5[i]=e225005[i];
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay5[i]=-3;
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay5[i]=e23901[i];
if e38329b[i]=2 and e38329k[i] le 0 then othpay5[i]=-3;
if e38329b[i]=2 and -4<e38329k[i]<0 then othpay5[i]=e225505[i];
if e38329b[i]=6 and e35600[i] le 0 then othpay5[i]=-3;
if e38329b[i]=6 and -4<e35600[i]<0 then othpay5[i]=e35600[i];

/*report non-hourly wage*/
if e34428[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay5[i]=e38329d[i];
if e34428[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
        othpay5[i]=(e38329d[i]*e38329k[i])/e34428[i];
if e34428[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
        then othpay5[i]=e38329d[i]/e34428[i];
if e34428[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(2*e34428[i]);
if e34428[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(4.3*e34428[i]);
if e34428[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
        othpay5[i]=e38329d[i]/(e35600[i]*e34428[i]);
if e34428[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(2.15*e34428[i]);
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
    if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e34428[i]>0 then
      othpay5[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e34428[i];
    if e38329i[i]=4 and e38329ib[i]>0 and e34428[i]>0 then othpay5[i]=(e38329d[i]/e38329ib[i])/e34428[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e34428[i]>0 then othpay5[i]=(e38329d[i]/e38329ib[i])/(e34428[i]*4.3);
  /* missing value */
    if e38329i[i] in (3,4,5) and -4<e34428[i]<=0 then othpay5[i]=-3;
 end;
```

```
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay5[i]=-3;
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay5[i]=e34428[i];

end;

/*without overtime at the beginning*/
if e3800b[i]=1 then do;

/* report hourly wage at the beginning*/
if e23901[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay5[i]=e38329d[i];
if e23901[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
        othpay5[i]=(e38329d[i]*e38329k[i])/e23901[i];
if e23901[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
        then othpay5[i]=e38329d[i]/e23901[i];
if e23901[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(2*e23901[i]);
if e23901[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(4.3*e23901[i]);
if e23901[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
        othpay5[i]=e38329d[i]/(e35600[i]*e23901[i]);
if e23901[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(2.15*e23901[i]);
if e38329b[i] in (9,14) then othpay5[i]=0;
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
   if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
     if e225301[i] in (1,2,4,5) and e225001[i]>0 and othpay5[i]>=0 then
        othpay5[i]=othpay5[i]/e225001[i]*e38329d[i];
     if e225301[i]=3 and e225001[i]>0 and e38329k[i]>0 and e225501[i]>0 and othpay5[i]>=0
       then othpay5[i]=othpay5[i]/(e225001[i]*e225501[i])*e38329d[i]*e38329k[i];
     if e225301[i]=3 and e38329k[i]<=0 then othpay5[i]=-3;
   end;
   else do; /*in all the other cases */
    if e38329i[i]=1 and e38329ib[i]>0 then othpay5[i]=(e38329d[i]/e38329ib[i])*60;
    if e38329i[i]=2 and e38329ib[i]>0 then othpay5[i]=e38329d[i]/e38329ib[i];
    if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e23901[i]>0 then
      othpay5[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e23901[i];
    if e38329i[i]=4 and e38329ib[i]>0 and e23901[i]>0 then othpay5[i]=(e38329d[i]/e38329ib[i])/e23901[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e23901[i]>0 then othpay5[i]=(e38329d[i]/e38329ib[i])/(e23901[i]*4.3);
  /* missing value */
    if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay5[i]=-3;
    if e38329i[i] in (3,4,5) and -4<e23901[i]<=0 then othpay5[i]=-3;
    if e38329i[i]=3 and e38329k[i]<0 then othpay5[i]=-3;
   end;
 end;

if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
        othpay5[i]=e225001[i];
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay5[i]=-3;
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay5[i]=e23901[i];
if e38329b[i]=2 and e38329k[i] le 0 then othpay5[i]=-3;
if e38329b[i]=2 and -4<e38329k[i]<0 then othpay5[i]=e225501[i];
if e38329b[i]=6 and e35600[i] le 0 then othpay5[i]=-3;
if e38329b[i]=6 and -4<e35600[i]<0 then othpay5[i]=e35600[i];

/*report non-hourly wage*/
if e34402[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay5[i]=e38329d[i];
if e34402[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
        othpay5[i]=(e38329d[i]*e38329k[i])/e34402[i];
```

```
  if e34402[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
          then othpay5[i]=e38329d[i]/e34402[i];
  if e34402[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(2*e34402[i]);
  if e34402[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(4.3*e34402[i]);
  if e34402[i]>0 and e38329b[i]=6 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(52*e34402[i]);
  if e34402[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(2.15*e34402[i]);
  if e38329b[i] in (12,13) and e38329d[i]>=o then do;
      if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e34402[i]>0 then
        othpay5[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e34402[i];
      if e38329i[i]=4 and e38329ib[i]>0 and e34402[i]>0 then othpay5[i]=(e38329d[i]/e38329ib[i])/e34402[i];
      if e38329i[i]=5 and e38329ib[i]>0 and e34402[i]>0 then othpay5[i]=(e38329d[i]/e38329ib[i])/(e34402[i]*4.3);
    /* missing value */
      if e38329i[i] in (3,4,5) and -4<e34402[i]<=0 then othpay5[i]=-3;
   end;

  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay5[i]=-3;
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          othpay5[i]=e34402[i];

  end;
 end;

 end;

/*** case iv. one compensation at the beginning, diff no. of hours. ***/
if e38329[i]=1 and e20700[i]=1 then do;

 /* for tips*/

 if e212002[i]=1 then do;
  /* without overtime at the beginning*/
  if e3800b[i]=0 then do;
  if e3800ff[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay1[i]=e38329d[i];
  if e3800ff[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
          othpay1[i]=(e38329d[i]*e38329k[i])/e3800ff[i];
  if e3800ff[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
          then othpay1[i]=e38329d[i]/e3800ff[i];
  if e3800ff[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(2*e3800ff[i]);
  if e3800ff[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(4.3*e3800ff[i]);
  if e3800ff[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
          othpay1[i]=e38329d[i]/(e35600[i]*e3800ff[i]);
  if e3800ff[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(2.15*e3800ff[i]);

  if e38329b[i] in (9,14) then othpay1[i]=0;
  if e38329b[i] in (12,13) and e38329d[i]>=o then do;
     if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
       if e225301[i] in (1,2,4,5) and e225001[i]>0 and othpay1[i]>=0 then
          othpay1[i]=othpay1[i]/e225001[i]*e38329d[i];
       if e225301[i]=3 and e225001[i]>0 and e38329k[i]>0 and e225501[i]>0 and othpay1[i]>=0
         then othpay1[i]=othpay1[i]/(e225001[i]*e225501[i])*e38329d[i]*e38329k[i];
       if e225301[i]=3 and e38329k[i]<=0 then othpay1[i]=-3;
     end;
     else do; /*in all the other cases */
      if e38329i[i]=1 and e38329ib[i]>0 then othpay1[i]=(e38329d[i]/e38329ib[i])*60;
      if e38329i[i]=2 and e38329ib[i]>0 then othpay1[i]=e38329d[i]/e38329ib[i];
      if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e3800ff[i]>0 then
       othpay1[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e3800ff[i];
```

```
    if e38329i[i]=4 and e38329ib[i]>0 and e3800ff[i]>0 then othpay1[i]=(e38329d[i]/e38329ib[i])/e3800ff[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e3800ff[i]>0 then othpay1[i]=(e38329d[i]/e38329ib[i])/(e3800ff[i]*4.3);
   /* missing value */
    if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay1[i]=-3;
    if e38329i[i] in (3,4,5) and -4<e3800ff[i]<=0 then othpay1[i]=-3;
    if e38329i[i]=3 and e38329k[i]<0 then othpay1[i]=-3;
   end;
 end;

if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
        othpay1[i]=e225001[i];
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e3800ff[i]=0 then othpay1[i]=-3;
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e3800ff[i]<0 then
        othpay1[i]=e3800ff[i];
if e38329b[i]=2 and e38329k[i] le 0 then othpay1[i]=-3;
if e38329b[i]=2 and -4<e38329k[i]<0 then othpay1[i]=e225501[i];
if e38329b[i]=6 and e35600[i] le 0 then othpay1[i]=-3;
if e38329b[i]=6 and -4<e35600[i]<0 then othpay1[i]=e35600[i];

end;

/* with overtime at the beginning*/
if e38102[i]=1 or e38102[i]=3 then do;
if e38103[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay1[i]=e38329d[i];
if e38103[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
        othpay1[i]=(e38329d[i]*e38329k[i])/e38103[i];
if e38103[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
        then othpay1[i]=e38329d[i]/e38103[i];
if e38103[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(2*e38103[i]);
if e38103[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(4.3*e38103[i]);
if e38103[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
        othpay1[i]=e38329d[i]/(e35600[i]*e38103[i]);
if e38103[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay1[i]=e38329d[i]/(2.15*e38103[i]);
if e38103[i]>0 and e38329b[i] in (9,7,12,13,14,15,16,17,18,19,21,22,23,24,25,26,28) then othpf1[i]=othpf1[i]+1;
if e38329b[i] in (9,14) then othpay1[i]=0;
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
   if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
      if e225301[i] in (1,2,4,5) and e225001[i]>0 and othpay1[i]>=0 then
         othpay1[i]=othpay1[i]/e225001[i]*e38329d[i];
      if e225301[i]=3 and e225001[i]>0 and e38329k[i]>0 and e225501[i]>0 and othpay1[i]>=0
        then othpay1[i]=othpay1[i]/(e225001[i]*e225501[i])*e38329d[i]*e38329k[i];
      if e225301[i]=3 and e38329k[i]<=0 then othpay1[i]=-3;
   end;
   else do; /*in all the other cases */
    if e38329i[i]=1 and e38329ib[i]>0 then othpay1[i]=(e38329d[i]/e38329ib[i])*60;
    if e38329i[i]=2 and e38329ib[i]>0 then othpay1[i]=e38329d[i]/e38329ib[i];
    if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e38103[i]>0 then
      othpay1[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e38103[i];
    if e38329i[i]=4 and e38329ib[i]>0 and e38103[i]>0 then othpay1[i]=(e38329d[i]/e38329ib[i])/e38103[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e38103[i]>0 then othpay1[i]=(e38329d[i]/e38329ib[i])/(e38103[i]*4.3);
   /* missing value */
    if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay1[i]=-3;
    if e38329i[i] in (3,4,5) and -4<e38103[i]<=0 then othpay1[i]=-3;
    if e38329i[i]=3 and e38329k[i]<0 then othpay1[i]=-3;
   end;
 end;
```

```
  if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
          othpay1[i]=e225001[i];
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e38103[i]=0 then othpay1[i]=-3;
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38103[i]<0 then
          othpay1[i]=e38103[i];
  if e38329b[i]=2 and e38329k[i] le 0 then othpay1[i]=-3;
  if e38329b[i]=2 and -4<e38329k[i]<0 then othpay1[i]=e225501[i];
  if e38329b[i]=6 and e35600[i] le 0 then othpay1[i]=-3;
  if e38329b[i]=6 and -4<e35600[i]<0 then othpay1[i]=e35600[i];

  end;
 end;

/*for commissions*/

 if e212003[i]=1 then do;
  /*without overtime at the beginning*/
  if e3800ff[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay2[i]=e38329d[i];
  if e3800ff[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
          othpay2[i]=(e38329d[i]*e38329k[i])/e3800ff[i];
  if e3800ff[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
          then othpay2[i]=e38329d[i]/e3800ff[i];
  if e3800ff[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(2*e3800ff[i]);
  if e3800ff[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(4.3*e3800ff[i]);
  if e3800ff[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
          othpay2[i]=e38329d[i]/(e35600[i]*e3800ff[i]);
  if e3800ff[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(2.15*e3800ff[i]);
  if e38329b[i] in (9,14) then othpay2[i]=0;
  if e38329b[i] in (12,13) and e38329d[i]>=o then do;
     if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
       if e225302[i] in (1,2,4,5) and e225002[i]>0 and othpay2[i]>=0 then
           othpay2[i]=othpay2[i]/e225002[i]*e38329d[i];
       if e225302[i]=3 and e225002[i]>0 and e38329k[i]>0 and e225502[i]>0 and othpay2[i]>=0
          then othpay2[i]=othpay2[i]/(e225002[i]*e225502[i])*e38329d[i]*e38329k[i];
       if e225302[i]=3 and e38329k[i]<=0 then othpay2[i]=-3;
     end;
     else do; /*in all the other cases */
      if e38329i[i]=1 and e38329ib[i]>0 then othpay2[i]=(e38329d[i]/e38329ib[i])*60;
      if e38329i[i]=2 and e38329ib[i]>0 then othpay2[i]=e38329d[i]/e38329ib[i];
      if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e3800ff[i]>0 then
        othpay2[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e3800ff[i];
      if e38329i[i]=4 and e38329ib[i]>0 and e3800ff[i]>0 then othpay2[i]=(e38329d[i]/e38329ib[i])/e3800ff[i];
      if e38329i[i]=5 and e38329ib[i]>0 and e3800ff[i]>0 then othpay2[i]=(e38329d[i]/e38329ib[i])/(e3800ff[i]*4.3);
    /* missing value */
      if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay2[i]=-3;
      if e38329i[i] in (3,4,5) and -4<e3800ff[i]<=0 then othpay2[i]=-3;
      if e38329i[i]=3 and e38329k[i]<0 then othpay2[i]=-3;
     end;
  end;

  if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
          othpay2[i]=e225002[i];
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e3800ff[i]=0 then othpay2[i]=-3;
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e3800ff[i]<0 then
          othpay2[i]=e3800ff[i];
  if e38329b[i]=2 and e38329k[i] le 0 then othpay2[i]=-3;
  if e38329b[i]=2 and -4<e38329k[i]<0 then othpay2[i]=e225502[i];
```

```
  if e38329b[i]=6 and e35600[i] le 0 then othpay2[i]=-3;
  if e38329b[i]=6 and -4<e35600[i]<0 then othpay2[i]=e35600[i];

  /* with overtime at the beginning*/
  if e38102[i]=1 or e38102[i]=3 then do;

  if e38103[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay2[i]=e38329d[i];
  if e38103[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
          othpay2[i]=(e38329d[i]*e38329k[i])/e38103[i];
  if e38103[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
          then othpay2[i]=e38329d[i]/e38103[i];
  if e38103[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(2*e38103[i]);
  if e38103[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(4.3*e38103[i]);
  if e38103[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
          othpay2[i]=e38329d[i]/(e35600[i]*e38103[i]);
  if e38103[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay2[i]=e38329d[i]/(2.15*e38103[i]);
  if e38103[i]>0 and e38329b[i] in (9,7,12,13,14,15,16,17,18,19,21,22,23,24,25,26,28) then othpf2[i]=othpf2[i]+1;
  if e38329b[i] in (9,14) then othpay2[i]=0;
  if e38329b[i] in (12,13) and e38329d[i]>=o then do;
     if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
       if e225302[i] in (1,2,4,5) and e225002[i]>0 and othpay2[i]>=0 then
          othpay2[i]=othpay2[i]/e225002[i]*e38329d[i];
       if e225302[i]=3 and e225002[i]>0 and e38329k[i]>0 and e225502[i]>0 and othpay2[i]>=0
         then othpay2[i]=othpay2[i]/(e225002[i]*e225502[i])*e38329d[i]*e38329k[i];
       if e225302[i]=3 and e38329k[i]<=0 then othpay2[i]=-3;
     end;
     else do; /*in all the other cases */
      if e38329i[i]=1 and e38329ib[i]>0 then othpay2[i]=(e38329d[i]/e38329ib[i])*60;
      if e38329i[i]=2 and e38329ib[i]>0 then othpay2[i]=e38329d[i]/e38329ib[i];
      if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e38103[i]>0 then
        othpay2[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e38103[i];
      if e38329i[i]=4 and e38329ib[i]>0 and e38103[i]>0 then othpay2[i]=(e38329d[i]/e38329ib[i])/e38103[i];
      if e38329i[i]=5 and e38329ib[i]>0 and e38103[i]>0 then othpay2[i]=(e38329d[i]/e38329ib[i])/(e38103[i]*4.3);
    /* missing value */
      if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay2[i]=-3;
      if e38329i[i] in (3,4,5) and -4<e38103[i]<=0 then othpay2[i]=-3;
      if e38329i[i]=3 and e38329k[i]<0 then othpay2[i]=-3;
     end;
   end;

  if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
          othpay2[i]=e225002[i];
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e38103[i]=0 then othpay2[i]=-3;
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38103[i]<0 then
          othpay2[i]=e38103[i];
  if e38329b[i]=2 and e38329k[i] le 0 then othpay2[i]=-3;
  if e38329b[i]=2 and -4<e38329k[i]<0 then othpay2[i]=e225502[i];
  if e38329b[i]=6 and e35600[i] le 0 then othpay2[i]=-3;
  if e38329b[i]=6 and -4<e35600[i]<0 then othpay2[i]=e35600[i];

  end;
 end;

/*for bonuses*/

 if e212004[i]=1 then do;
  /* without overtime at the beginning*/
```

```
if e3800ff[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay3[i]=e38329d[i];
if e3800ff[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
        othpay3[i]=(e38329d[i]*e38329k[i])/e3800ff[i];
if e3800ff[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
        then othpay3[i]=e38329d[i]/e3800ff[i];
if e3800ff[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(2*e3800ff[i]);
if e3800ff[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(4.3*e3800ff[i]);
if e3800ff[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
        othpay3[i]=e38329d[i]/(e35600[i]*e3800ff[i]);
if e3800ff[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(2.15*e3800ff[i]);
if e38329b[i] in (9,14) then othpay3[i]=0;
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
   if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
     if e225303[i] in (1,2,4,5) and e225003[i]>0 and othpay3[i]>=0 then
        othpay3[i]=othpay3[i]/e225003[i]*e38329d[i];
     if e225303[i]=3 and e225003[i]>0 and e38329k[i]>0 and e225503[i]>0 and othpay3[i]>=0
       then othpay3[i]=othpay3[i]/(e225003[i]*e225503[i])*e38329d[i]*e38329k[i];
     if e225303[i]=3 and e38329k[i]<=0 then othpay3[i]=-3;
   end;
   else do; /*in all the other cases */
    if e38329i[i]=1 and e38329ib[i]>0 then othpay3[i]=(e38329d[i]/e38329ib[i])*60;
    if e38329i[i]=2 and e38329ib[i]>0 then othpay3[i]=e38329d[i]/e38329ib[i];
    if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e3800ff[i]>0 then
      othpay3[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e3800ff[i];
    if e38329i[i]=4 and e38329ib[i]>0 and e3800ff[i]>0 then othpay3[i]=(e38329d[i]/e38329ib[i])/e3800ff[i];
    if e38329i[i]=5 and e38329ib[i]>0 and e3800ff[i]>0 then othpay3[i]=(e38329d[i]/e38329ib[i])/(e3800ff[i]*4.3);
  /* missing value */
    if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay3[i]=-3;
    if e38329i[i] in (3,4,5) and -4<e3800ff[i]<=0 then othpay3[i]=-3;
    if e38329i[i]=3 and e38329k[i]<0 then othpay3[i]=-3;
   end;
 end;

if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
        othpay3[i]=e225003[i];
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e3800ff[i]=0 then othpay3[i]=-3;
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e3800ff[i]<0 then
        othpay3[i]=e3800ff[i];
if e38329b[i]=2 and e38329k[i] le 0 then othpay3[i]=-3;
if e38329b[i]=2 and -4<e38329k[i]<0 then othpay3[i]=e225503[i];
if e38329b[i]=6 and e35600[i] le 0 then othpay3[i]=-3;
if e38329b[i]=6 and -4<e35600[i]<0 then othpay3[i]=e35600[i];

/*with overtime at the beginning*/
if e38102[i]=1 or e38102[i]=3 then do;
if e38103[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay3[i]=e38329d[i];
if e38103[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
        othpay3[i]=(e38329d[i]*e38329k[i])/e38103[i];
if e38103[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
        then othpay3[i]=e38329d[i]/e38103[i];
if e38103[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(2*e38103[i]);
if e38103[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(4.3*e38103[i]);
if e38103[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
        othpay3[i]=e38329d[i]/(e35600[i]*e38103[i]);
if e38103[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay3[i]=e38329d[i]/(2.15*e38103[i]);
if e38329b[i] in (9,14) then othpay3[i]=0;
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
```

```
    if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
      if e225303[i] in (1,2,4,5) and e225003[i]>0 and othpay3[i]>=0 then
          othpay3[i]=othpay3[i]/e225003[i]*e38329d[i];
      if e225303[i]=3 and e225003[i]>0 and e38329k[i]>0 and e225503[i]>0 and othpay3[i]>=0
         then othpay3[i]=othpay3[i]/(e225003[i]*e225503[i])*e38329d[i]*e38329k[i];
      if e225303[i]=3 and e38329k[i]<=0 then othpay3[i]=-3;
    end;
    else do; /*in all the other cases */
     if e38329i[i]=1 and e38329ib[i]>0 then othpay3[i]=(e38329d[i]/e38329ib[i])*60;
     if e38329i[i]=2 and e38329ib[i]>0 then othpay3[i]=e38329d[i]/e38329ib[i];
     if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e38103[i]>0 then
       othpay3[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e38103[i];
     if e38329i[i]=4 and e38329ib[i]>0 and e38103[i]>0 then othpay3[i]=(e38329d[i]/e38329ib[i])/e38103[i];
     if e38329i[i]=5 and e38329ib[i]>0 and e38103[i]>0 then othpay3[i]=(e38329d[i]/e38329ib[i])/(e38103[i]*4.3);
   /* missing value */
     if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay3[i]=-3;
     if e38329i[i] in (3,4,5) and -4<e38103[i]<=0 then othpay3[i]=-3;
     if e38329i[i]=3 and e38329k[i]<0 then othpay3[i]=-3;
    end;
  end;

  if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
          othpay3[i]=e225003[i];
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e38103[i]=0 then othpay3[i]=-3;
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38103[i]<0 then
          othpay3[i]=e38103[i];
  if e38329b[i]=2 and e38329k[i] le 0 then othpay3[i]=-3;
  if e38329b[i]=2 and -4<e38329k[i]<0 then othpay3[i]=e225503[i];
  if e38329b[i]=6 and e35600[i] le 0 then othpay3[i]=-3;
  if e38329b[i]=6 and -4<e35600[i]<0 then othpay3[i]=e35600[i];

  end;
 end;

/*for incentive pay*/

if e212005[i]=1 then do;
  /*without overtime at the beginning*/
  if e3800ff[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay4[i]=e38329d[i];
  if e3800ff[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
          othpay4[i]=(e38329d[i]*e38329k[i])/e3800ff[i];
  if e3800ff[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
          then othpay4[i]=e38329d[i]/e3800ff[i];
  if e3800ff[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(2*e3800ff[i]);
  if e3800ff[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(4.3*e3800ff[i]);
  if e3800ff[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
          othpay4[i]=e38329d[i]/(e35600[i]*e3800ff[i]);
  if e3800ff[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(2.15*e3800ff[i]);
  if e38329b[i] in (9,14) then othpay4[i]=0;
  if e38329b[i] in (12,13) and e38329d[i]>=o then do;
    if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
      if e225304[i] in (1,2,4,5) and e225004[i]>0 and othpay4[i]>=0 then
          othpay4[i]=othpay4[i]/e225004[i]*e38329d[i];
      if e225304[i]=3 and e225004[i]>0 and e38329k[i]>0 and e225504[i]>0 and othpay4[i]>=0
         then othpay4[i]=othpay4[i]/(e225004[i]*e225504[i])*e38329d[i]*e38329k[i];
      if e225304[i]=3 and e38329k[i]<=0 then othpay4[i]=-3;
    end;
```

```
    else do; /*in all the other cases */
     if e38329i[i]=1 and e38329ib[i]>0 then othpay4[i]=(e38329d[i]/e38329ib[i])*60;
     if e38329i[i]=2 and e38329ib[i]>0 then othpay4[i]=e38329d[i]/e38329ib[i];
     if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e3800ff[i]>0 then
       othpay4[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e3800ff[i];
     if e38329i[i]=4 and e38329ib[i]>0 and e3800ff[i]>0 then othpay4[i]=(e38329d[i]/e38329ib[i])/e3800ff[i];
     if e38329i[i]=5 and e38329ib[i]>0 and e3800ff[i]>0 then othpay4[i]=(e38329d[i]/e38329ib[i])/(e3800ff[i]*4.3);
   /* missing value */
     if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay4[i]=-3;
     if e38329i[i] in (3,4,5) and -4<e3800ff[i]<=0 then othpay4[i]=-3;
     if e38329i[i]=3 and e38329k[i]<0 then othpay4[i]=-3;
    end;
 end;

if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
          othpay4[i]=e225004[i];
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e3800ff[i]=0 then othpay4[i]=-3;
if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e3800ff[i]<0 then
          othpay4[i]=e3800ff[i];
if e38329b[i]=2 and e38329k[i] le 0 then othpay4[i]=-3;
if e38329b[i]=2 and -4<e38329k[i]<0 then othpay4[i]=e225504[i];
if e38329b[i]=6 and e35600[i] le 0 then othpay4[i]=-3;
if e38329b[i]=6 and -4<e35600[i]<0 then othpay4[i]=e35600[i];

/* with overtime at the beginning*/
if e38102[i]=1 or e38102[i]=3 then do;
if e38103[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay4[i]=e38329d[i];
if e38103[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
          othpay4[i]=(e38329d[i]*e38329k[i])/e38103[i];
if e38103[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
          then othpay4[i]=e38329d[i]/e38103[i];
if e38103[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(2*e38103[i]);
if e38103[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(4.3*e38103[i]);
if e38103[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
          othpay4[i]=e38329d[i]/(e35600[i]*e38103[i]);
if e38103[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay4[i]=e38329d[i]/(2.15*e38103[i]);
if e38329b[i] in (9,14) then othpay4[i]=0;
if e38329b[i] in (12,13) and e38329d[i]>=o then do;
    if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
       if e225304[i] in (1,2,4,5) and e225004[i]>0 and othpay4[i]>=0 then
          othpay4[i]=othpay4[i]/e225004[i]*e38329d[i];
       if e225304[i]=3 and e225004[i]>0 and e38329k[i]>0 and e225504[i]>0 and othpay4[i]>=0
         then othpay4[i]=othpay4[i]/(e225004[i]*e225504[i])*e38329d[i]*e38329k[i];
       if e225304[i]=3 and e38329k[i]<=0 then othpay4[i]=-3;
    end;
    else do; /*in all the other cases */
     if e38329i[i]=1 and e38329ib[i]>0 then othpay4[i]=(e38329d[i]/e38329ib[i])*60;
     if e38329i[i]=2 and e38329ib[i]>0 then othpay4[i]=e38329d[i]/e38329ib[i];
     if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e38103[i]>0 then
       othpay4[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e38103[i];
     if e38329i[i]=4 and e38329ib[i]>0 and e38103[i]>0 then othpay4[i]=(e38329d[i]/e38329ib[i])/e38103[i];
     if e38329i[i]=5 and e38329ib[i]>0 and e38103[i]>0 then othpay4[i]=(e38329d[i]/e38329ib[i])/(e38103[i]*4.3);
   /* missing value */
     if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay4[i]=-3;
     if e38329i[i] in (3,4,5) and -4<e38103[i]<=0 then othpay4[i]=-3;
     if e38329i[i]=3 and e38329k[i]<0 then othpay4[i]=-3;
    end;
```

```
   end;

  if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
          othpay4[i]=e225004[i];
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e38103[i]=0 then othpay4[i]=-3;
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38103[i]<0 then
          othpay4[i]=e38103[i];
  if e38329b[i]=2 and e38329k[i] le 0 then othpay4[i]=-3;
  if e38329b[i]=2 and -4<e38329k[i]<0 then othpay4[i]=e225504[i];
  if e38329b[i]=6 and e35600[i] le 0 then othpay4[i]=-3;
  if e38329b[i]=6 and -4<e35600[i]<0 then othpay4[i]=e35600[i];

  end;
end;

/*for others*/

 if e212006[i]=1 then do;
  /* without overtime at the beginning*/
  if e3800ff[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay5[i]=e38329d[i];
  if e3800ff[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
          othpay5[i]=(e38329d[i]*e38329k[i])/e3800ff[i];
  if e3800ff[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
          then othpay5[i]=e38329d[i]/e3800ff[i];
  if e3800ff[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(2*e3800ff[i]);
  if e3800ff[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(4.3*e3800ff[i]);
  if e3800ff[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
          othpay5[i]=e38329d[i]/(e35600[i]*e3800ff[i]);
  if e3800ff[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(2.15*e3800ff[i]);
  if e38329b[i] in (9,14) then othpay5[i]=0;
  if e38329b[i] in (12,13) and e38329d[i]>=o then do;
     if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
       if e225305[i] in (1,2,4,5) and e225005[i]>0 and othpay5[i]>=0 then
          othpay5[i]=othpay5[i]/e225005[i]*e38329d[i];
       if e225305[i]=3 and e225005[i]>0 and e38329k[i]>0 and e225505[i]>0 and othpay5[i]>=0
         then othpay5[i]=othpay5[i]/(e225005[i]*e225505[i])*e38329d[i]*e38329k[i];
       if e225305[i]=3 and e38329k[i]<=0 then othpay5[i]=-3;
     end;
     else do; /*in all the other cases */
      if e38329i[i]=1 and e38329ib[i]>0 then othpay5[i]=(e38329d[i]/e38329ib[i])*60;
      if e38329i[i]=2 and e38329ib[i]>0 then othpay5[i]=e38329d[i]/e38329ib[i];
      if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e3800ff[i]>0 then
        othpay5[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e3800ff[i];
      if e38329i[i]=4 and e38329ib[i]>0 and e3800ff[i]>0 then othpay5[i]=(e38329d[i]/e38329ib[i])/e3800ff[i];
      if e38329i[i]=5 and e38329ib[i]>0 and e3800ff[i]>0 then othpay5[i]=(e38329d[i]/e38329ib[i])/(e3800ff[i]*4.3);
    /* missing value */
      if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay5[i]=-3;
      if e38329i[i] in (3,4,5) and -4<e3800ff[i]<=0 then othpay5[i]=-3;
      if e38329i[i]=3 and e38329k[i]<0 then othpay5[i]=-3;
     end;
   end;

  if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
          othpay5[i]=e225005[i];
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e3800ff[i]=0 then othpay5[i]=-3;
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e3800ff[i]<0 then
          othpay5[i]=e3800ff[i];
```

```
  if e38329b[i]=2 and e38329k[i] le 0 then othpay5[i]=-3;
  if e38329b[i]=2 and -4<e38329k[i]<0 then othpay5[i]=e225505[i];
  if e38329b[i]=6 and e35600[i] le 0 then othpay5[i]=-3;
  if e38329b[i]=6 and -4<e35600[i]<0 then othpay5[i]=e35600[i];

  /* with overtime at the beginning*/
  if e38102[i]=1 or e38102[i]=3 then do;
  if e38103[i]>0 and e38329b[i]=1 and e38329d[i]>=0 then othpay5[i]=e38329d[i];
  if e38103[i]>0 and e38329b[i]=2 and e38329k[i]>0 and e38329d[i]>=0 then
          othpay5[i]=(e38329d[i]*e38329k[i])/e38103[i];
  if e38103[i]>0 and e38329b[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e38329d[i]>=0
          then othpay5[i]=e38329d[i]/e38103[i];
  if e38103[i]>0 and e38329b[i]=4 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(2*e38103[i]);
  if e38103[i]>0 and e38329b[i]=5 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(4.3*e38103[i]);
  if e38103[i]>0 and e38329b[i]=6 and e38329d[i]>=0 and e35600[i]>0 then
          othpay5[i]=e38329d[i]/(e35600[i]*e38103[i]);
  if e38103[i]>0 and e38329b[i]=8 and e38329d[i]>=0 then othpay5[i]=e38329d[i]/(2.15*e38103[i]);
  if e38329b[i] in (9,14) then othpay5[i]=0;
  if e38329b[i] in (12,13) and e38329d[i]>=o then do;
     if e38329g[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
       if e225305[i] in (1,2,4,5) and e225005[i]>0 and othpay5[i]>=0 then
          othpay5[i]=othpay5[i]/e225005[i]*e38329d[i];
       if e225305[i]=3 and e225005[i]>0 and e38329k[i]>0 and e225505[i]>0 and othpay5[i]>=0
         then othpay5[i]=othpay5[i]/(e225005[i]*e225505[i])*e38329d[i]*e38329k[i];
       if e225305[i]=3 and e38329k[i]<=0 then othpay5[i]=-3;
     end;
     else do; /*in all the other cases */
      if e38329i[i]=1 and e38329ib[i]>0 then othpay5[i]=(e38329d[i]/e38329ib[i])*60;
      if e38329i[i]=2 and e38329ib[i]>0 then othpay5[i]=e38329d[i]/e38329ib[i];
      if e38329i[i]=3 and e38329ib[i]>0 and e38329k[i]>=0 and e38103[i]>0 then
        othpay5[i]=(e38329d[i]/e38329ib[i])*e38329k[i]/e38103[i];
      if e38329i[i]=4 and e38329ib[i]>0 and e38103[i]>0 then othpay5[i]=(e38329d[i]/e38329ib[i])/e38103[i];
      if e38329i[i]=5 and e38329ib[i]>0 and e38103[i]>0 then othpay5[i]=(e38329d[i]/e38329ib[i])/(e38103[i]*4.3);
     /* missing value */
      if -4<e38329i[i]<0 or -4<e38329ib[i]<=0 then othpay5[i]=-3;
      if e38329i[i] in (3,4,5) and -4<e38103[i]<=0 then othpay5[i]=-3;
      if e38329i[i]=3 and e38329k[i]<0 then othpay5[i]=-3;
     end;
   end;

  if e38329b[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38329d[i]<0 then
          othpay5[i]=e225005[i];
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e38103[i]=0 then othpay5[i]=-3;
  if e38329b[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38103[i]<0 then
          othpay5[i]=e38103[i];
  if e38329b[i]=2 and e38329k[i] le 0 then othpay5[i]=-3;
  if e38329b[i]=2 and -4<e38329k[i]<0 then othpay5[i]=e225505[i];
  if e38329b[i]=6 and e35600[i] le 0 then othpay5[i]=-3;
  if e38329b[i]=6 and -4<e35600[i]<0 then othpay5[i]=e35600[i];

  end;
 end;
end;

/*** case v. more then one compensation at the beginning, same no. of hours. ***/
if e38330[i]=1 and e20700[i]=1 then do;
```

```
/* for tips*/

 /*with overtime at the beginning*/
 if e38102[i]>0 and e38102[i] ne 1 and e38102[i] ne 3 then do;

 /*report hourly wage at the beginning*/
 if e23901[i]>0 and e384071[i]=1 and e384161[i]>=0 then othpay1[i]=e384161[i];
 if e23901[i]>0 and e384071[i]=2 and e38416k1[i]>0 and e384161[i]>=0 then
          othpay1[i]=(e384161[i]*e38416k1[i])/e23901[i];
 if e23901[i]>0 and e384071[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384161[i]>=0
          then othpay1[i]=e384161[i]/e23901[i];
 if e23901[i]>0 and e384071[i]=4 and e384161[i]>=0 then othpay1[i]=e384161[i]/(2*e23901[i]);
 if e23901[i]>0 and e384071[i]=5 and e384161[i]>=0 then othpay1[i]=e384161[i]/(4.3*e23901[i]);
 if e23901[i]>0 and e384071[i]=6 and e384161[i]>=0 and e35600[i]>0 then
          othpay1[i]=e384161[i]/(e35600[i]*e23901[i]);
 if e23901[i]>0 and e384071[i]=8 and e384161[i]>=0 then othpay1[i]=e384161[i]/(2.15*e23901[i]);
 if e384071[i] in (9,14) then othpay1[i]=0;
 if e384071[i] in (12,13) and e384161[i]>=o then do;
    if e38416g1[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
       if e225301[i] in (1,2,4,5) and e225001[i]>0 and othpay1[i]>=0 then
          othpay1[i]=othpay1[i]/e225001[i]*e384161[i];
       if e225301[i]=3 and e225001[i]>0 and e38416k1[i]>0 and e225501[i]>0 and othpay1[i]>=0
         then othpay1[i]=othpay1[i]/(e225001[i]*e225501[i])*e384161[i]*e38416k1[i];
       if e225301[i]=3 and e38416k1[i]<=0 then othpay1[i]=-3;
    end;
    else do; /*in all the other cases */
     if e38416i1[i]=1 and e38416ib1[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])*60;
     if e38416i1[i]=2 and e38416ib1[i]>0 then othpay1[i]=e384161[i]/e38416ib1[i];
     if e38416i1[i]=3 and e38416ib1[i]>0 and e38416k1[i]>=0 and e23901[i]>0 then
       othpay1[i]=(e384161[i]/e38416ib1[i])*e38416k1[i]/e23901[i];
     if e38416i1[i]=4 and e38416ib1[i]>0 and e23901[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])/e23901[i];
     if e38416i1[i]=5 and e38416ib1[i]>0 and e23901[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])/(e23901[i]*4.3);
   /* missing value */
     if -4<e38416i1[i]<0 or -4<e38416ib1[i]<=0 then othpay1[i]=-3;
     if e38416i1[i] in (3,4,5) and -4<e23901[i]<=0 then othpay1[i]=-3;
     if e38416i1[i]=3 and e38416k1[i]<0 then othpay1[i]=-3;
    end;
  end;

 if e384071[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384161[i]<0 then
          othpay1[i]=e225001[i];
 if e384071[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay1[i]=-3;
 if e384071[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
          othpay1[i]=e23901[i];
 if e384071[i]=2 and e38416k1[i] le 0 then othpay1[i]=-3;
 if e384071[i]=2 and -4<e38416k1[i]<0 then othpay1[i]=e225501[i];
 if e384071[i]=6 and e35600[i] le 0 then othpay1[i]=-3;
 if e384071[i]=6 and -4<e35600[i]<0 then othpay1[i]=e35600[i];

 /*report non-hourly wage*/
 if e34428[i]>0 and e384071[i]=1 and e384161[i]>=0 then othpay1[i]=e384161[i];
 if e34428[i]>0 and e384071[i]=2 and e38416k1[i]>0 and e384161[i]>=0 then
          othpay1[i]=(e384161[i]*e38416k1[i])/e34428[i];
 if e34428[i]>0 and e384071[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384161[i]>=0
          then othpay1[i]=e384161[i]/e34428[i];
 if e34428[i]>0 and e384071[i]=4 and e384161[i]>=0 then othpay1[i]=e384161[i]/(2*e34428[i]);
 if e34428[i]>0 and e384071[i]=5 and e384161[i]>=0 then othpay1[i]=e384161[i]/(4.3*e34428[i]);
```

```
if e34428[i]>0 and e384071[i]=6 and e384161[i]>=0 and e35600[i]>0 then
        othpay1[i]=e384161[i]/(e35600[i]*e34428[i]);
if e34428[i]>0 and e384071[i]=8 and e384161[i]>=0 then othpay1[i]=e384161[i]/(2.15*e34428[i]);
if e34428[i]>0 and e384071[i] in (9,7,12,13,14,15,16,17,18,19,21,22,23,24,25,26,28) then othpf1[i]=othpf1[i]+1;
if e384071[i] in (12,13) and e384161[i]>=o then do;
    if e38416i1[i]=3 and e38416ib1[i]>0 and e38416k1[i]>=0 and e34428[i]>0 then
      othpay1[i]=(e384161[i]/e38416ib1[i])*e38416k1[i]/e34428[i];
    if e38416i1[i]=4 and e38416ib1[i]>0 and e34428[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])/e34428[i];
    if e38416i1[i]=5 and e38416ib1[i]>0 and e34428[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])/(e34428[i]*4.3);
  /* missing value */
    if e38416i1[i] in (3,4,5) and -4<e34428[i]<=0 then othpay1[i]=-3;
   end;

if e384071[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay1[i]=-3;
if e384071[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay1[i]=e34428[i];

end;

/*without overtime at the beginning*/
if e3800b[i]=1 then do;

/*report hourly wage at the beginning*/
if e23901[i]>0 and e384071[i]=1  and e384161[i]>=0 then othpay1[i]=e384161[i];
if e23901[i]>0 and e384071[i]=2  and e38416k1[i]>0 and e384161[i]>=0 then
        othpay1[i]=(e384161[i]*e38416k1[i])/e23901[i];
if e23901[i]>0 and e384071[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2)  and e384161[i]>=0
        then othpay1[i]=e384161[i]/e23901[i];
if e23901[i]>0 and e384071[i]=4  and e384161[i]>=0 then othpay1[i]=e384161[i]/(2*e23901[i]);
if e23901[i]>0 and e384071[i]=5  and e384161[i]>=0 then othpay1[i]=e384161[i]/(4.3*e23901[i]);
if e23901[i]>0 and e384071[i]=6  and e384161[i]>=0 and e35600[i]>0 then
        othpay1[i]=e384161[i]/(e35600[i]*e23901[i]);
if e23901[i]>0 and e384071[i]=8  and e384161[i]>=0 then othpay1[i]=e384161[i]/(2.15*e23901[i]);
if e384071[i] in (9,14) then othpay1[i]=0;
if e384071[i] in (12,13) and e384161[i]>=o then do;
   if e38416g1[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
      if e225301[i] in (1,2,4,5) and e225001[i]>0 and othpay1[i]>=0 then
         othpay1[i]=othpay1[i]/e225001[i]*e384161[i];
      if e225301[i]=3 and e225001[i]>0 and e38416k1[i]>0 and e225501[i]>0 and othpay1[i]>=0
        then othpay1[i]=othpay1[i]/(e225001[i]*e225501[i])*e384161[i]*e38416k1[i];
      if e225301[i]=3 and e38416k1[i]<=0 then othpay1[i]=-3;
   end;
   else do; /*in all the other cases */
    if e38416i1[i]=1 and e38416ib1[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])*60;
    if e38416i1[i]=2 and e38416ib1[i]>0 then othpay1[i]=e384161[i]/e38416ib1[i];
    if e38416i1[i]=3 and e38416ib1[i]>0 and e38416k1[i]>=0 and e23901[i]>0 then
      othpay1[i]=(e384161[i]/e38416ib1[i])*e38416k1[i]/e23901[i];
    if e38416i1[i]=4 and e38416ib1[i]>0 and e23901[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])/e23901[i];
    if e38416i1[i]=5 and e38416ib1[i]>0 and e23901[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])/(e23901[i]*4.3);
  /* missing value */
    if -4<e38416i1[i]<0 or -4<e38416ib1[i]<=0 then othpay1[i]=-3;
    if e38416i1[i] in (3,4,5) and -4<e23901[i]<=0 then othpay1[i]=-3;
    if e38416i1[i]=3 and e38416k1[i]<0 then othpay1[i]=-3;
   end;
 end;
```

```
if e384071[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384161[i]<0 then
        othpay1[i]=e225001[i];
if e384071[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay1[i]=-3;
if e384071[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay1[i]=e23901[i];
if e384071[i]=2 and e38416k1[i] le 0 then othpay1[i]=-3;
if e384071[i]=2 and -4<e38416k1[i]<0 then othpay1[i]=e225501[i];
if e384071[i]=6 and e35600[i] le 0 then othpay1[i]=-3;
if e384071[i]=6 and -4<e35600[i]<0 then othpay1[i]=e35600[i];

/*report non-hourly wage*/
if e34402[i]>0 and e384071[i]=1 and e384161[i]>=0 then othpay1[i]=e384161[i];
if e34402[i]>0 and e384071[i]=2 and e38416k1[i]>0 and e384161[i]>=0 then
        othpay1[i]=(e384161[i]*e38416k1[i])/e34402[i];
if e34402[i]>0 and e384071[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384161[i]>=0
        then othpay1[i]=e384161[i]/e34402[i];
if e34402[i]>0 and e384071[i]=4 and e384161[i]>=0 then othpay1[i]=e384161[i]/(2*e34402[i]);
if e34402[i]>0 and e384071[i]=5 and e384161[i]>=0 then othpay1[i]=e384161[i]/(4.3*e34402[i]);
if e34402[i]>0 and e384071[i]=6 and e384161[i]>=0 and e35600[i]>0 then
        othpay1[i]=e384161[i]/(e35600[i]*e34402[i]);
if e34402[i]>0 and e384071[i]=8 and e384161[i]>=0 then othpay1[i]=e384161[i]/(2.15*e34402[i]);
if e34402[i]>0 and e384071[i] in (9,7,12,13,14,15,16,17,18,19,21,22,23,24,25,26,28) then othpf1[i]=othpf1[i]+1;
if e384071[i] in (12,13) and e384161[i]>=o then do;
    if e38416i1[i]=3 and e38416ib1[i]>0 and e38416k1[i]>=0 and e34402[i]>0 then
      othpay1[i]=(e384161[i]/e38416ib1[i])*e38416k1[i]/e34402[i];
    if e38416i1[i]=4 and e38416ib1[i]>0 and e34402[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])/e34402[i];
    if e38416i1[i]=5 and e38416ib1[i]>0 and e34402[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])/(e34402[i]*4.3);
   /* missing value */
    if e38416i1[i] in (3,4,5) and -4<e34402[i]<=0 then othpay1[i]=-3;
   end;

if e384071[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay1[i]=-3;
if e384071[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
        othpay1[i]=e34402[i];

end;

/*for commissions*/

/*with overtime at the beginning*/
if e38102[i]>0 and e38102[i] ne 1 and e38102[i] ne 3 then do;

 /* report hourly wage at the beginning*/
if e23901[i]>0 and e384072[i]=1 and e384162[i]>=0 then othpay2[i]=e384162[i];
if e23901[i]>0 and e384072[i]=2 and e38416k2[i]>0 and e384162[i]>=0 then
        othpay2[i]=(e384162[i]*e38416k2[i])/e23901[i];
if e23901[i]>0 and e384072[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384162[i]>=0
        then othpay2[i]=e384162[i]/e23901[i];
if e23901[i]>0 and e384072[i]=4 and e384162[i]>=0 then othpay2[i]=e384162[i]/(2*e23901[i]);
if e23901[i]>0 and e384072[i]=5 and e384162[i]>=0 then othpay2[i]=e384162[i]/(4.3*e23901[i]);
if e23901[i]>0 and e384072[i]=6 and e384162[i]>=0 and e35600[i]>0 then
        othpay2[i]=e384162[i]/(e35600[i]*e23901[i]);
if e23901[i]>0 and e384072[i]=8 and e384162[i]>=0 then othpay2[i]=e384162[i]/(2.15*e23901[i]);
if e384072[i] in (9,14) then othpay2[i]=0;
if e384072[i] in (12,13) and e384162[i]>=o then do;
   if e38416g2[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
```

```
      if e225302[i] in (1,2,4,5) and e225002[i]>0 and othpay2[i]>=0 then
          othpay2[i]=othpay2[i]/e225002[i]*e384162[i];
      if e225302[i]=3 and e225002[i]>0 and e38416k2[i]>0 and e225502[i]>0 and othpay2[i]>=0
         then othpay2[i]=othpay2[i]/(e225002[i]*e225502[i])*e384162[i]*e38416k2[i];
      if e225302[i]=3 and e38416k2[i]<=0 then othpay2[i]=-3;
    end;
    else do; /*in all the other cases */
     if e38416i2[i]=1 and e38416ib2[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])*60;
     if e38416i2[i]=2 and e38416ib2[i]>0 then othpay2[i]=e384162[i]/e38416ib2[i];
     if e38416i2[i]=3 and e38416ib2[i]>0 and e38416k2[i]>=0 and e23901[i]>0 then
       othpay2[i]=(e384162[i]/e38416ib2[i])*e38416k2[i]/e23901[i];
     if e38416i2[i]=4 and e38416ib2[i]>0 and e23901[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])/e23901[i];
     if e38416i2[i]=5 and e38416ib2[i]>0 and e23901[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])/(e23901[i]*4.3);
   /* missing value */
     if -4<e38416i2[i]<0 or -4<e38416ib2[i]<=0 then othpay2[i]=-3;
     if e38416i2[i] in (3,4,5) and -4<e23901[i]<=0 then othpay2[i]=-3;
     if e38416i2[i]=3 and e38416k2[i]<0 then othpay2[i]=-3;
    end;
 end;

if e384072[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384162[i]<0 then
        othpay2[i]=e225002[i];
if e384072[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay2[i]=-3;
if e384072[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay2[i]=e23901[i];
if e384072[i]=2 and e38416k2[i] le 0 then othpay2[i]=-3;
if e384072[i]=2 and -4<e38416k2[i]<0 then othpay2[i]=e225502[i];
if e384072[i]=6 and e35600[i] le 0 then othpay2[i]=-3;
if e384072[i]=6 and -4<e35600[i]<0 then othpay2[i]=e35600[i];

/* report non-hourly wage*/
if e34428[i]>0 and e384072[i]=1 and e384162[i]>=0 then othpay2[i]=e384162[i];
if e34428[i]>0 and e384072[i]=2 and e38416k2[i]>0 and e384162[i]>=0 then
        othpay2[i]=(e384162[i]*e38416k2[i])/e34428[i];
if e34428[i]>0 and e384072[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384162[i]>=0
        then othpay2[i]=e384162[i]/e34428[i];
if e34428[i]>0 and e384072[i]=4 and e384162[i]>=0 then othpay2[i]=e384162[i]/(2*e34428[i]);
if e34428[i]>0 and e384072[i]=5 and e384162[i]>=0 then othpay2[i]=e384162[i]/(4.3*e34428[i]);
if e34428[i]>0 and e384072[i]=6 and e384162[i]>=0 and e35600[i]>0 then
        othpay2[i]=e384162[i]/(e35600[i]*e34428[i]);
if e34428[i]>0 and e384072[i]=8 and e384162[i]>=0 then othpay2[i]=e384162[i]/(2.15*e34428[i]);
if e384072[i] in (12,13) and e384162[i]>=o then do;
    if e38416i2[i]=3 and e38416ib2[i]>0 and e38416k2[i]>=0 and e34428[i]>0 then
      othpay2[i]=(e384162[i]/e38416ib2[i])*e38416k2[i]/e34428[i];
    if e38416i2[i]=4 and e38416ib2[i]>0 and e34428[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])/e34428[i];
    if e38416i2[i]=5 and e38416ib2[i]>0 and e34428[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])/(e34428[i]*4.3);
  /* missing value */
    if e38416i2[i] in (3,4,5) and -4<e34428[i]<=0 then othpay2[i]=-3;
   end;

if e384072[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay2[i]=-3;
if e384072[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay2[i]=e34428[i];

end;

/*without overtime at the beginning*/
```

```
if e3800b[i]=1 then do;

/* report hourly wage at the beginning*/
if e23901[i]>0 and e384072[i]=1 and e384162[i]>=0 then othpay2[i]=e384162[i];
if e23901[i]>0 and e384072[i]=2 and e38416k2[i]>0 and e384162[i]>=0 then
        othpay2[i]=(e384162[i]*e38416k2[i])/e23901[i];
if e23901[i]>0 and e384072[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384162[i]>=0
        then othpay2[i]=e384162[i]/e23901[i];
if e23901[i]>0 and e384072[i]=4 and e384162[i]>=0 then othpay2[i]=e384162[i]/(2*e23901[i]);
if e23901[i]>0 and e384072[i]=5 and e384162[i]>=0 then othpay2[i]=e384162[i]/(4.3*e23901[i]);
if e23901[i]>0 and e384072[i]=6 and e384162[i]>=0 and e35600[i]>0 then
        othpay2[i]=e384162[i]/(e35600[i]*e23901[i]);
if e23901[i]>0 and e384072[i]=8 and e384162[i]>=0 then othpay2[i]=e384162[i]/(2.15*e23901[i]);
if e384072[i] in (9,14) then othpay2[i]=0;
if e384072[i] in (12,13) and e384162[i]>=o then do;
   if e38416g2[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
     if e225302[i] in (1,2,4,5) and e225002[i]>0 and othpay2[i]>=0 then
        othpay2[i]=othpay2[i]/e225002[i]*e384162[i];
     if e225302[i]=3 and e225002[i]>0 and e38416k2[i]>0 and e225502[i]>0 and othpay2[i]>=0
       then othpay2[i]=othpay2[i]/(e225002[i]*e225502[i])*e384162[i]*e38416k2[i];
     if e225302[i]=3 and e38416k2[i]<=0 then othpay2[i]=-3;
   end;
   else do; /*in all the other cases */
    if e38416i2[i]=1 and e38416ib2[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])*60;
    if e38416i2[i]=2 and e38416ib2[i]>0 then othpay2[i]=e384162[i]/e38416ib2[i];
    if e38416i2[i]=3 and e38416ib2[i]>0 and e38416k2[i]>=0 and e23901[i]>0 then
      othpay2[i]=(e384162[i]/e38416ib2[i])*e38416k2[i]/e23901[i];
    if e38416i2[i]=4 and e38416ib2[i]>0 and e23901[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])/e23901[i];
    if e38416i2[i]=5 and e38416ib2[i]>0 and e23901[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])/(e23901[i]*4.3);
  /* missing value */
    if -4<e38416i2[i]<0 or -4<e38416ib2[i]<=0 then othpay2[i]=-3;
    if e38416i2[i] in (3,4,5) and -4<e23901[i]<=0 then othpay2[i]=-3;
    if e38416i2[i]=3 and e38416k2[i]<0 then othpay2[i]=-3;
   end;
 end;

if e384072[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384162[i]<0 then
        othpay2[i]=e225002[i];
if e384072[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay2[i]=-3;
if e384072[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay2[i]=e23901[i];
if e384072[i]=2 and e38416k2[i] le 0 then othpay2[i]=-3;
if e384072[i]=2 and -4<e38416k2[i]<0 then othpay2[i]=e225502[i];
if e384072[i]=6 and e35600[i] le 0 then othpay2[i]=-3;
if e384072[i]=6 and -4<e35600[i]<0 then othpay2[i]=e35600[i];

/* report non-hourly wage*/
if e34402[i]>0 and e384072[i]=1  and e384162[i]>=0 then othpay2[i]=e384162[i];
if e34402[i]>0 and e384072[i]=2  and e38416k2[i]>0 and e384162[i]>=0 then
        othpay2[i]=(e384162[i]*e38416k2[i])/e34402[i];
if e34402[i]>0 and e384072[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2)  and e384162[i]>=0
        then othpay2[i]=e384162[i]/e34402[i];
if e34402[i]>0 and e384072[i]=4  and e384162[i]>=0 then othpay2[i]=e384162[i]/(2*e34402[i]);
if e34402[i]>0 and e384072[i]=5  and e384162[i]>=0 then othpay2[i]=e384162[i]/(4.3*e34402[i]);
if e34402[i]>0 and e384072[i]=6  and e384162[i]>=0 and e35600[i]>0 then
        othpay2[i]=e384162[i]/(e35600[i]*e34402[i]);
if e34402[i]>0 and e384072[i]=8  and e384162[i]>=0 then othpay2[i]=e384162[i]/(2.15*e34402[i]);
```

```
 if e34402[i]>0 and e384072[i] in (9,7,12,13,14,15,16,17,18,19,21,22,23,24,25,26,28) then othpf2[i]=othpf2[i]+1;
 if e384072[i] in (12,13) and e384162[i]>=o then do;
     if e38416i2[i]=3 and e38416ib2[i]>0 and e38416k2[i]>=0 and e34402[i]>0 then
       othpay2[i]=(e384162[i]/e38416ib2[i])*e38416k2[i]/e34402[i];
     if e38416i2[i]=4 and e38416ib2[i]>0 and e34402[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])/e34402[i];
     if e38416i2[i]=5 and e38416ib2[i]>0 and e34402[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])/(e34402[i]*4.3);
   /* missing value */
     if e38416i2[i] in (3,4,5) and -4<e34402[i]<=0 then othpay2[i]=-3;
    end;

 if e384072[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay2[i]=-3;
 if e384072[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          othpay2[i]=e34402[i];

 end;

/*for bonuses*/

 /*with overtime at the beginning*/
 if e38102[i]>0 and e38102[i] ne 1 and e38102[i] ne 3 then do;

 /* report hourly wage at the beginning*/
 if e23901[i]>0 and e384073[i]=1 and e384163[i]>=0 then othpay3[i]=e384163[i];
 if e23901[i]>0 and e384073[i]=2 and e38416k3[i]>0 and e384163[i]>=0 then
          othpay3[i]=(e384163[i]*e38416k3[i])/e23901[i];
 if e23901[i]>0 and e384073[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384163[i]>=0
          then othpay3[i]=e384163[i]/e23901[i];
 if e23901[i]>0 and e384073[i]=4 and e384163[i]>=0 then othpay3[i]=e384163[i]/(2*e23901[i]);
 if e23901[i]>0 and e384073[i]=5 and e384163[i]>=0 then othpay3[i]=e384163[i]/(4.3*e23901[i]);
 if e23901[i]>0 and e384073[i]=6 and e384163[i]>=0 and e35600[i]>0 then
          othpay3[i]=e384163[i]/(e35600[i]*e23901[i]);
 if e23901[i]>0 and e384073[i]=8 and e384163[i]>=0 then othpay3[i]=e384163[i]/(2.15*e23901[i]);
  if e384073[i] in (9,14) then othpay3[i]=0;
 if e384073[i] in (12,13) and e384163[i]>=o then do;
    if e38416g3[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
      if e225303[i] in (1,2,4,5) and e225003[i]>0 and othpay3[i]>=0 then
          othpay3[i]=othpay3[i]/e225003[i]*e384163[i];
      if e225303[i]=3 and e225003[i]>0 and e38416k3[i]>0 and e225503[i]>0 and othpay3[i]>=0
         then othpay3[i]=othpay3[i]/(e225003[i]*e225503[i])*e384163[i]*e38416k3[i];
      if e225303[i]=3 and e38416k3[i]<=0 then othpay3[i]=-3;
    end;
    else do; /*in all the other cases */
     if e38416i3[i]=1 and e38416ib3[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])*60;
     if e38416i3[i]=2 and e38416ib3[i]>0 then othpay3[i]=e384163[i]/e38416ib3[i];
     if e38416i3[i]=3 and e38416ib3[i]>0 and e38416k3[i]>=0 and e23901[i]>0 then
       othpay3[i]=(e384163[i]/e38416ib3[i])*e38416k3[i]/e23901[i];
     if e38416i3[i]=4 and e38416ib3[i]>0 and e23901[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])/e23901[i];
     if e38416i3[i]=5 and e38416ib3[i]>0 and e23901[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])/(e23901[i]*4.3);
   /* missing value */
     if -4<e38416i3[i]<0 or -4<e38416ib3[i]<=0 then othpay3[i]=-3;
     if e38416i3[i] in (3,4,5) and -4<e23901[i]<=0 then othpay3[i]=-3;
     if e38416i3[i]=3 and e38416k3[i]<0 then othpay3[i]=-3;
    end;
  end;

 if e384073[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384163[i]<0 then
          othpay3[i]=e225003[i];
```

```
if e384073[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay3[i]=-3;
if e384073[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay3[i]=e23901[i];
if e384073[i]=2 and e38416k3[i] le 0 then othpay3[i]=-3;
if e384073[i]=2 and -4<e38416k3[i]<0 then othpay3[i]=e225503[i];
if e384073[i]=6 and e35600[i] le 0 then othpay3[i]=-3;
if e384073[i]=6 and -4<e35600[i]<0 then othpay3[i]=e35600[i];

/*report non-hurly wage*/
if e34428[i]>0 and e384073[i]=1 and e384163[i]>=0 then othpay3[i]=e384163[i];
if e34428[i]>0 and e384073[i]=2 and e38416k3[i]>0 and e384163[i]>=0 then
        othpay3[i]=(e384163[i]*e38416k3[i])/e34428[i];
if e34428[i]>0 and e384073[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384163[i]>=0
        then othpay3[i]=e384163[i]/e34428[i];
if e34428[i]>0 and e384073[i]=4 and e384163[i]>=0 then othpay3[i]=e384163[i]/(2*e34428[i]);
if e34428[i]>0 and e384073[i]=5 and e384163[i]>=0 then othpay3[i]=e384163[i]/(4.3*e34428[i]);
if e34428[i]>0 and e384073[i]=6 and e384163[i]>=0 and e35600[i]>0 then
        othpay3[i]=e384163[i]/(e35600[i]*e34428[i]);
if e34428[i]>0 and e384073[i]=8 and e384163[i]>=0 then othpay3[i]=e384163[i]/(2.15*e34428[i]);
if e384073[i] in (12,13) and e384163[i]>=o then do;
    if e38416i3[i]=3 and e38416ib3[i]>0 and e38416k3[i]>=0 and e34428[i]>0 then
      othpay3[i]=(e384163[i]/e38416ib3[i])*e38416k3[i]/e34428[i];
    if e38416i3[i]=4 and e38416ib3[i]>0 and e34428[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])/e34428[i];
    if e38416i3[i]=5 and e38416ib3[i]>0 and e34428[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])/(e34428[i]*4.3);
  /* missing value */
    if e38416i3[i] in (3,4,5) and -4<e34428[i]<=0 then othpay3[i]=-3;
   end;

if e384073[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay3[i]=-3;
if e384073[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay3[i]=e34428[i];

end;

/*without overtime at the beginning*/
if e3800b[i]=1 then do;

/* report hourly wage at the beginning*/
if e23901[i]>0 and e384073[i]=1 and e384163[i]>=0 then othpay3[i]=e384163[i];
if e23901[i]>0 and e384073[i]=2 and e38416k3[i]>0 and e384163[i]>=0 then
        othpay3[i]=(e384163[i]*e38416k3[i])/e23901[i];
if e23901[i]>0 and e384073[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384163[i]>=0
        then othpay3[i]=e384163[i]/e23901[i];
if e23901[i]>0 and e384073[i]=4 and e384163[i]>=0 then othpay3[i]=e384163[i]/(2*e23901[i]);
if e23901[i]>0 and e384073[i]=5 and e384163[i]>=0 then othpay3[i]=e384163[i]/(4.3*e23901[i]);
if e23901[i]>0 and e384073[i]=6 and e384163[i]>=0 and e35600[i]>0 then
        othpay3[i]=e384163[i]/(e35600[i]*e23901[i]);
if e23901[i]>0 and e384073[i]=8 and e384163[i]>=0 then othpay3[i]=e384163[i]/(2.15*e23901[i]);
if e384073[i] in (9,14) then othpay3[i]=0;
if e384073[i] in (12,13) and e384163[i]>=o then do;
   if e38416g3[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
      if e225303[i] in (1,2,4,5) and e225003[i]>0 and othpay3[i]>=0 then
         othpay3[i]=othpay3[i]/e225003[i]*e384163[i];
      if e225303[i]=3 and e225003[i]>0 and e38416k3[i]>0 and e225503[i]>0 and othpay3[i]>=0
         then othpay3[i]=othpay3[i]/(e225003[i]*e225503[i])*e384163[i]*e38416k3[i];
      if e225303[i]=3 and e38416k3[i]<=0 then othpay3[i]=-3;
   end;
```

```
    else do; /*in all the other cases */
     if e38416i3[i]=1 and e38416ib3[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])*60;
     if e38416i3[i]=2 and e38416ib3[i]>0 then othpay3[i]=e384163[i]/e38416ib3[i];
     if e38416i3[i]=3 and e38416ib3[i]>0 and e38416k3[i]>=0 and e23901[i]>0 then
       othpay3[i]=(e384163[i]/e38416ib3[i])*e38416k3[i]/e23901[i];
     if e38416i3[i]=4 and e38416ib3[i]>0 and e23901[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])/e23901[i];
     if e38416i3[i]=5 and e38416ib3[i]>0 and e23901[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])/(e23901[i]*4.3);
    /* missing value */
     if -4<e38416i3[i]<0 or -4<e38416ib3[i]<=0 then othpay3[i]=-3;
     if e38416i3[i] in (3,4,5) and -4<e23901[i]<=0 then othpay3[i]=-3;
     if e38416i3[i]=3 and e38416k3[i]<0 then othpay3[i]=-3;
    end;
  end;

 if e384073[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384163[i]<0 then
          othpay3[i]=e225003[i];
 if e384073[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay3[i]=-3;
 if e384073[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
          othpay3[i]=e23901[i];
 if e384073[i]=2 and e38416k3[i] le 0 then othpay3[i]=-3;
 if e384073[i]=2 and -4<e38416k3[i]<0 then othpay3[i]=e225503[i];
 if e384073[i]=6 and e35600[i] le 0 then othpay3[i]=-3;
 if e384073[i]=6 and -4<e35600[i]<0 then othpay3[i]=e35600[i];

 /*report non-hurly wage*/
 if e34402[i]>0 and e384073[i]=1 and e384163[i]>=0 then othpay3[i]=e384163[i];
 if e34402[i]>0 and e384073[i]=2 and e38416k3[i]>0 and e384163[i]>=0 then
          othpay3[i]=(e384163[i]*e38416k3[i])/e34402[i];
 if e34402[i]>0 and e384073[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384163[i]>=0
          then othpay3[i]=e384163[i]/e34402[i];
 if e34402[i]>0 and e384073[i]=4 and e384163[i]>=0 then othpay3[i]=e384163[i]/(2*e34402[i]);
 if e34402[i]>0 and e384073[i]=5 and e384163[i]>=0 then othpay3[i]=e384163[i]/(4.3*e34402[i]);
 if e34402[i]>0 and e384073[i]=6 and e384163[i]>=0 and e35600[i]>0 then
          othpay3[i]=e384163[i]/(e35600[i]*e34402[i]);
 if e34402[i]>0 and e384073[i]=8 and e384163[i]>=0 then othpay3[i]=e384163[i]/(2.15*e34402[i]);
 if e34402[i]>0 and e384073[i] in (9,7,12,13,14,15,16,17,18,19,21,22,23,24,25,26,28) then othpf3[i]=othpf3[i]+1;
 if e384073[i] in (12,13) and e384163[i]>=o then do;
     if e38416i3[i]=3 and e38416ib3[i]>0 and e38416k3[i]>=0 and e34402[i]>0 then
       othpay3[i]=(e384163[i]/e38416ib3[i])*e38416k3[i]/e34402[i];
     if e38416i3[i]=4 and e38416ib3[i]>0 and e34402[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])/e34402[i];
     if e38416i3[i]=5 and e38416ib3[i]>0 and e34402[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])/(e34402[i]*4.3);
    /* missing value */
     if e38416i3[i] in (3,4,5) and -4<e34402[i]<=0 then othpay3[i]=-3;
    end;

 if e384073[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay3[i]=-3;
 if e384073[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          othpay3[i]=e34402[i];

 end;

/*for incentive pay*/

 /*with overtime at the beginning*/
 if e38102[i]>0 and e38102[i] ne 1 and e38102[i] ne 3 then do;

  /*report hourly wage at the beginning*/
```

```
if e23901[i]>0 and e384074[i]=1 and e384164[i]>=0 then othpay4[i]=e384164[i];
if e23901[i]>0 and e384074[i]=2 and e38416k4[i]>0 and e384164[i]>=0 then
        othpay4[i]=(e384164[i]*e38416k4[i])/e23901[i];
if e23901[i]>0 and e384074[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384164[i]>=0
        then othpay4[i]=e384164[i]/e23901[i];
if e23901[i]>0 and e384074[i]=4 and e384164[i]>=0 then othpay4[i]=e384164[i]/(2*e23901[i]);
if e23901[i]>0 and e384074[i]=5 and e384164[i]>=0 then othpay4[i]=e384164[i]/(4.3*e23901[i]);
if e23901[i]>0 and e384074[i]=6 and e384164[i]>=0 and e35600[i]>0 then
        othpay4[i]=e384164[i]/(e35600[i]*e23901[i]);
if e23901[i]>0 and e384074[i]=8 and e384164[i]>=0 then othpay4[i]=e384164[i]/(2.15*e23901[i]);
 if e384074[i] in (9,14) then othpay4[i]=0;
if e384074[i] in (12,13) and e384164[i]>=o then do;
   if e38416g4[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
     if e225304[i] in (1,2,4,5) and e225004[i]>0 and othpay4[i]>=0 then
        othpay4[i]=othpay4[i]/e225004[i]*e384164[i];
     if e225304[i]=3 and e225004[i]>0 and e38416k4[i]>0 and e225504[i]>0 and othpay4[i]>=0
       then othpay4[i]=othpay4[i]/(e225004[i]*e225504[i])*e384164[i]*e38416k4[i];
     if e225304[i]=3 and e38416k4[i]<=0 then othpay4[i]=-3;
   end;
   else do; /*in all the other cases */
    if e38416i4[i]=1 and e38416ib4[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])*60;
    if e38416i4[i]=2 and e38416ib4[i]>0 then othpay4[i]=e384164[i]/e38416ib4[i];
    if e38416i4[i]=3 and e38416ib4[i]>0 and e38416k4[i]>=0 and e23901[i]>0 then
      othpay4[i]=(e384164[i]/e38416ib4[i])*e38416k4[i]/e23901[i];
    if e38416i4[i]=4 and e38416ib4[i]>0 and e23901[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])/e23901[i];
    if e38416i4[i]=5 and e38416ib4[i]>0 and e23901[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])/(e23901[i]*4.3);
  /* missing value */
    if -4<e38416i4[i]<0 or -4<e38416ib4[i]<=0 then othpay4[i]=-3;
    if e38416i4[i] in (3,4,5) and -4<e23901[i]<=0 then othpay4[i]=-3;
    if e38416i4[i]=3 and e38416k4[i]<0 then othpay4[i]=-3;
   end;
 end;

if e384074[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384164[i]<0 then
        othpay4[i]=e225004[i];
if e384074[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay4[i]=-3;
if e384074[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay4[i]=e23901[i];
if e384074[i]=2 and e38416k4[i] le 0 then othpay4[i]=-3;
if e384074[i]=2 and -4<e38416k4[i]<0 then othpay4[i]=e225504[i];
if e384074[i]=6 and e35600[i] le 0 then othpay4[i]=-3;
if e384074[i]=6 and -4<e35600[i]<0 then othpay4[i]=e35600[i];

/*report non-hourly wage*/
if e34428[i]>0 and e384074[i]=1 and e384164[i]>=0 then othpay4[i]=e384164[i];
if e34428[i]>0 and e384074[i]=2 and e38416k4[i]>0 and e384164[i]>=0 then
        othpay4[i]=(e384164[i]*e38416k4[i])/e34428[i];
if e34428[i]>0 and e384074[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384164[i]>=0
        then othpay4[i]=e384164[i]/e34428[i];
if e34428[i]>0 and e384074[i]=4 and e384164[i]>=0 then othpay4[i]=e384164[i]/(2*e34428[i]);
if e34428[i]>0 and e384074[i]=5 and e384164[i]>=0 then othpay4[i]=e384164[i]/(4.3*e34428[i]);
if e34428[i]>0 and e384074[i]=6 and e384164[i]>=0 and e35600[i]>0 then
        othpay4[i]=e384164[i]/(e35600[i]*e34428[i]);
if e34428[i]>0 and e384074[i]=8 and e384164[i]>=0 then othpay4[i]=e384164[i]/(2.15*e34428[i]);
 if e384074[i] in (12,13) and e384164[i]>=o then do;
    if e38416i4[i]=3 and e38416ib4[i]>0 and e38416k4[i]>=0 and e34428[i]>0 then
      othpay4[i]=(e384164[i]/e38416ib4[i])*e38416k4[i]/e34428[i];
```

```
    if e38416i4[i]=4 and e38416ib4[i]>0 and e34428[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])/e34428[i];
    if e38416i4[i]=5 and e38416ib4[i]>0 and e34428[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])/(e34428[i]*4.3);
  /* missing value */
    if e38416i4[i] in (3,4,5) and -4<e34428[i]<=0 then othpay4[i]=-3;
   end;

if e384074[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay4[i]=-3;
if e384074[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay4[i]=e34428[i];

end;

/*without overtime at the beginning*/
if e3800b[i]=1 then do;

 /*report hourly wage at the beginning*/
if e23901[i]>0 and e384074[i]=1 and e384164[i]>=0 then othpay4[i]=e384164[i];
if e23901[i]>0 and e384074[i]=2 and e38416k4[i]>0 and e384164[i]>=0 then
        othpay4[i]=(e384164[i]*e38416k4[i])/e23901[i];
if e23901[i]>0 and e384074[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384164[i]>=0
        then othpay4[i]=e384164[i]/e23901[i];
if e23901[i]>0 and e384074[i]=4 and e384164[i]>=0 then othpay4[i]=e384164[i]/(2*e23901[i]);
if e23901[i]>0 and e384074[i]=5 and e384164[i]>=0 then othpay4[i]=e384164[i]/(4.3*e23901[i]);
if e23901[i]>0 and e384074[i]=6 and e384164[i]>=0 and e35600[i]>0 then
        othpay4[i]=e384164[i]/(e35600[i]*e23901[i]);
if e23901[i]>0 and e384074[i]=8 and e384164[i]>=0 then othpay4[i]=e384164[i]/(2.15*e23901[i]);
if e384074[i] in (9,14) then othpay4[i]=0;
if e384074[i] in (12,13) and e384164[i]>=o then do;
   if e38416g4[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
     if e225304[i] in (1,2,4,5) and e225004[i]>0 and othpay4[i]>=0 then
        othpay4[i]=othpay4[i]/e225004[i]*e384164[i];
     if e225304[i]=3 and e225004[i]>0 and e38416k4[i]>0 and e225504[i]>0 and othpay4[i]>=0
       then othpay4[i]=othpay4[i]/(e225004[i]*e225504[i])*e384164[i]*e38416k4[i];
     if e225304[i]=3 and e38416k4[i]<=0 then othpay4[i]=-3;
   end;
   else do; /*in all the other cases */
    if e38416i4[i]=1 and e38416ib4[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])*60;
    if e38416i4[i]=2 and e38416ib4[i]>0 then othpay4[i]=e384164[i]/e38416ib4[i];
    if e38416i4[i]=3 and e38416ib4[i]>0 and e38416k4[i]>=0 and e23901[i]>0 then
      othpay4[i]=(e384164[i]/e38416ib4[i])*e38416k4[i]/e23901[i];
    if e38416i4[i]=4 and e38416ib4[i]>0 and e23901[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])/e23901[i];
    if e38416i4[i]=5 and e38416ib4[i]>0 and e23901[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])/(e23901[i]*4.3);
  /* missing value */
    if -4<e38416i4[i]<0 or -4<e38416ib4[i]<=0 then othpay4[i]=-3;
    if e38416i4[i] in (3,4,5) and -4<e23901[i]<=0 then othpay4[i]=-3;
    if e38416i4[i]=3 and e38416k4[i]<0 then othpay4[i]=-3;
   end;
 end;

if e384074[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384164[i]<0 then
        othpay4[i]=e225004[i];
if e384074[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay4[i]=-3;
if e384074[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay4[i]=e23901[i];
if e384074[i]=2 and e38416k4[i] le 0 then othpay4[i]=-3;
if e384074[i]=2 and -4<e38416k4[i]<0 then othpay4[i]=e225504[i];
if e384074[i]=6 and e35600[i] le 0 then othpay4[i]=-3;
```

```
if e384074[i]=6 and -4<e35600[i]<0 then othpay4[i]=e35600[i];

/*report non-hourly wage*/
if e34402[i]>0 and e384074[i]=1 and e384164[i]>=0 then othpay4[i]=e384164[i];
if e34402[i]>0 and e384074[i]=2 and e38416k4[i]>0 and e384164[i]>=0 then
        othpay4[i]=(e384164[i]*e38416k4[i])/e34402[i];
if e34402[i]>0 and e384074[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384164[i]>=0
        then othpay4[i]=e384164[i]/e34402[i];
if e34402[i]>0 and e384074[i]=4 and e384164[i]>=0 then othpay4[i]=e384164[i]/(2*e34402[i]);
if e34402[i]>0 and e384074[i]=5 and e384164[i]>=0 then othpay4[i]=e384164[i]/(4.3*e34402[i]);
if e34402[i]>0 and e384074[i]=6 and e384164[i]>=0 and e35600[i]>0 then
        othpay4[i]=e384164[i]/(e35600[i]*e34402[i]);
if e34402[i]>0 and e384074[i]=8 and e384164[i]>=0 then othpay4[i]=e384164[i]/(2.15*e34402[i]);
if e34402[i]>0 and e384074[i] in (9,7,12,13,114,15,16,17,18,19,21,22,23,24,25,26,28) then othpf4[i]=othpf4[i]+1;
if e384074[i] in (12,13) and e384164[i]>=o then do;
    if e38416i4[i]=3 and e38416ib4[i]>0 and e38416k4[i]>=0 and e34402[i]>0 then
      othpay4[i]=(e384164[i]/e38416ib4[i])*e38416k4[i]/e34402[i];
    if e38416i4[i]=4 and e38416ib4[i]>0 and e34402[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])/e34402[i];
    if e38416i4[i]=5 and e38416ib4[i]>0 and e34402[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])/(e34402[i]*4.3);
   /* missing value */
    if e38416i4[i] in (3,4,5) and -4<e34402[i]<=0 then othpay4[i]=-3;
   end;

if e384074[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay4[i]=-3;
if e384074[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
        othpay4[i]=e34402[i];

end;

/*for others*/

/*with overtime at the beginning*/
if e38102[i]>0 and e38102[i] ne 1 and e38102[i] ne 3 then do;

/* report hourly wage at the beginning*/
if e23901[i]>0 and e384075[i]=1 and e384165[i]>=0 then othpay5[i]=e384165[i];
if e23901[i]>0 and e384075[i]=2 and e38416k5[i]>0 and e384165[i]>=0 then
        othpay5[i]=(e384165[i]*e38416k5[i])/e23901[i];
if e23901[i]>0 and e384075[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384165[i]>=0
        then othpay5[i]=e384165[i]/e23901[i];
if e23901[i]>0 and e384075[i]=4 and e384165[i]>=0 then othpay5[i]=e384165[i]/(2*e23901[i]);
if e23901[i]>0 and e384075[i]=5 and e384165[i]>=0 then othpay5[i]=e384165[i]/(4.3*e23901[i]);
if e23901[i]>0 and e384075[i]=6 and e384165[i]>=0 and e35600[i]>0 then
        othpay5[i]=e384165[i]/(e35600[i]*e23901[i]);
if e23901[i]>0 and e384075[i]=8 and e384165[i]>=0 then othpay5[i]=e384165[i]/(2.15*e23901[i]);
 if e384075[i] in (9,14) then othpay5[i]=0;
if e384075[i] in (12,13) and e384165[i]>=o then do;
   if e38416g5[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
     if e225305[i] in (1,2,4,5) and e225005[i]>0 and othpay5[i]>=0 then
        othpay5[i]=othpay5[i]/e225005[i]*e384165[i];
     if e225305[i]=3 and e225005[i]>0 and e38416k5[i]>0 and e225505[i]>0 and othpay5[i]>=0
       then othpay5[i]=othpay5[i]/(e225005[i]*e225505[i])*e384165[i]*e38416k5[i];
     if e225305[i]=3 and e38416k5[i]<=0 then othpay5[i]=-3;
   end;
   else do; /*in all the other cases */
    if e38416i5[i]=1 and e38416ib5[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])*60;
    if e38416i5[i]=2 and e38416ib5[i]>0 then othpay5[i]=e384165[i]/e38416ib5[i];
```

```
    if e38416i5[i]=3 and e38416ib5[i]>0 and e38416k5[i]>=0 and e23901[i]>0 then
      othpay5[i]=(e384165[i]/e38416ib5[i])*e38416k5[i]/e23901[i];
    if e38416i5[i]=4 and e38416ib5[i]>0 and e23901[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])/e23901[i];
    if e38416i5[i]=5 and e38416ib5[i]>0 and e23901[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])/(e23901[i]*4.3);
  /* missing value */
    if -4<e38416i5[i]<0 or -4<e38416ib5[i]<=0 then othpay5[i]=-3;
    if e38416i5[i] in (3,4,5) and -4<e23901[i]<=0 then othpay5[i]=-3;
    if e38416i5[i]=3 and e38416k5[i]<0 then othpay5[i]=-3;
   end;
 end;

if e384075[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384165[i]<0 then
        othpay5[i]=e225005[i];
if e384075[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay5[i]=-3;
if e384075[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay5[i]=e23901[i];
if e384075[i]=2 and e38416k5[i] le 0 then othpay5[i]=-3;
if e384075[i]=2 and -4<e38416k5[i]<0 then othpay5[i]=e225505[i];
if e384075[i]=6 and e35600[i] le 0 then othpay5[i]=-3;
if e384075[i]=6 and -4<e35600[i]<0 then othpay5[i]=e35600[i];

/*report non-hourly wage*/
if e34428[i]>0 and e384075[i]=1 and e384165[i]>=0 then othpay5[i]=e384165[i];
if e34428[i]>0 and e384075[i]=2 and e38416k5[i]>0 and e384165[i]>=0 then
        othpay5[i]=(e384165[i]*e38416k5[i])/e34428[i];
if e34428[i]>0 and e384075[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384165[i]>=0
        then othpay5[i]=e384165[i]/e34428[i];
if e34428[i]>0 and e384075[i]=4 and e384165[i]>=0 then othpay5[i]=e384165[i]/(2*e34428[i]);
if e34428[i]>0 and e384075[i]=5 and e384165[i]>=0 then othpay5[i]=e384165[i]/(4.3*e34428[i]);
if e34428[i]>0 and e384075[i]=6 and e384165[i]>=0 and e35600[i]>0 then
        othpay5[i]=e384165[i]/(e35600[i]*e34428[i]);
if e34428[i]>0 and e384075[i]=8 and e384165[i]>=0 then othpay5[i]=e384165[i]/(2.15*e34428[i]);
if e384075[i] in (12,13) and e384165[i]>=o then do;
    if e38416i5[i]=3 and e38416ib5[i]>0 and e38416k5[i]>=0 and e34428[i]>0 then
      othpay5[i]=(e384165[i]/e38416ib5[i])*e38416k5[i]/e34428[i];
    if e38416i5[i]=4 and e38416ib5[i]>0 and e34428[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])/e34428[i];
    if e38416i5[i]=5 and e38416ib5[i]>0 and e34428[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])/(e34428[i]*4.3);
  /* missing value */
    if e38416i5[i] in (3,4,5) and -4<e34428[i]<=0 then othpay5[i]=-3;
   end;

if e384075[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34428[i]=0 then othpay5[i]=-3;
if e384075[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34428[i]<0 then
        othpay5[i]=e34428[i];

end;

/*without overtime at the beginning*/
if e3800b[i]=1 then do;

/* report hourly wage at the beginning*/
if e23901[i]>0 and e384075[i]=1 and e384165[i]>=0 then othpay5[i]=e384165[i];
if e23901[i]>0 and e384075[i]=2 and e38416k5[i]>0 and e384165[i]>=0 then
        othpay5[i]=(e384165[i]*e38416k5[i])/e23901[i];
if e23901[i]>0 and e384075[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384165[i]>=0
        then othpay5[i]=e384165[i]/e23901[i];
if e23901[i]>0 and e384075[i]=4 and e384165[i]>=0 then othpay5[i]=e384165[i]/(2*e23901[i]);
```

```
if e23901[i]>0 and e384075[i]=5 and e384165[i]>=0 then othpay5[i]=e384165[i]/(4.3*e23901[i]);
if e23901[i]>0 and e384075[i]=6 and e384165[i]>=0 and e35600[i]>0 then
        othpay5[i]=e384165[i]/(e35600[i]*e23901[i]);
if e23901[i]>0 and e384075[i]=8 and e384165[i]>=0 then othpay5[i]=e384165[i]/(2.15*e23901[i]);
if e384075[i] in (9,14) then othpay5[i]=0;
if e384075[i] in (12,13) and e384165[i]>=o then do;
   if e38416g5[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
     if e225305[i] in (1,2,4,5) and e225005[i]>0 and othpay5[i]>=0 then
        othpay5[i]=othpay5[i]/e225005[i]*e384165[i];
     if e225305[i]=3 and e225005[i]>0 and e38416k5[i]>0 and e225505[i]>0 and othpay5[i]>=0
       then othpay5[i]=othpay5[i]/(e225005[i]*e225505[i])*e384165[i]*e38416k5[i];
     if e225305[i]=3 and e38416k5[i]<=0 then othpay5[i]=-3;
   end;
   else do; /*in all the other cases */
    if e38416i5[i]=1 and e38416ib5[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])*60;
    if e38416i5[i]=2 and e38416ib5[i]>0 then othpay5[i]=e384165[i]/e38416ib5[i];
    if e38416i5[i]=3 and e38416ib5[i]>0 and e38416k5[i]>=0 and e23901[i]>0 then
      othpay5[i]=(e384165[i]/e38416ib5[i])*e38416k5[i]/e23901[i];
    if e38416i5[i]=4 and e38416ib5[i]>0 and e23901[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])/e23901[i];
    if e38416i5[i]=5 and e38416ib5[i]>0 and e23901[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])/(e23901[i]*4.3);
  /* missing value */
    if -4<e38416i5[i]<0 or -4<e38416ib5[i]<=0 then othpay5[i]=-3;
    if e38416i5[i] in (3,4,5) and -4<e23901[i]<=0 then othpay5[i]=-3;
    if e38416i5[i]=3 and e38416k5[i]<0 then othpay5[i]=-3;
   end;
 end;

if e384075[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384165[i]<0 then
        othpay5[i]=e225005[i];
if e384075[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e23901[i]=0 then othpay5[i]=-3;
if e384075[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e23901[i]<0 then
        othpay5[i]=e23901[i];
if e384075[i]=2 and e38416k5[i] le 0 then othpay5[i]=-3;
if e384075[i]=2 and -4<e38416k5[i]<0 then othpay5[i]=e225505[i];
if e384075[i]=6 and e35600[i] le 0 then othpay5[i]=-3;
if e384075[i]=6 and -4<e35600[i]<0 then othpay5[i]=e35600[i];

/*report non-hourly wage*/
if e34402[i]>0 and e384075[i]=1 and e384165[i]>=0 then othpay5[i]=e384165[i];
if e34402[i]>0 and e384075[i]=2 and e38416k5[i]>0 and e384165[i]>=0 then
        othpay5[i]=(e384165[i]*e38416k5[i])/e34402[i];
if e34402[i]>0 and e384075[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384165[i]>=0
        then othpay5[i]=e384165[i]/e34402[i];
if e34402[i]>0 and e384075[i]=4 and e384165[i]>=0 then othpay5[i]=e384165[i]/(2*e34402[i]);
if e34402[i]>0 and e384075[i]=5 and e384165[i]>=0 then othpay5[i]=e384165[i]/(4.3*e34402[i]);
if e34402[i]>0 and e384075[i]=6 and e384165[i]>=0 and e35600[i]>0 then
        othpay5[i]=e384165[i]/(e35600[i]*e34402[i]);
if e34402[i]>0 and e384075[i]=8 and e384165[i]>=0 then othpay5[i]=e384165[i]/(2.15*e34402[i]);
if e34402[i]>0 and e384075[i] in (9,7,12,13,14,15,16,17,18,19,21,22,23,24,25,26,28) then othpf5[i]=othpf5[i]+1;
if e384075[i] in (12,13) and e384165[i]>=o then do;
    if e38416i5[i]=3 and e38416ib5[i]>0 and e38416k5[i]>=0 and e34402[i]>0 then
      othpay5[i]=(e384165[i]/e38416ib5[i])*e38416k5[i]/e34402[i];
    if e38416i5[i]=4 and e38416ib5[i]>0 and e34402[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])/e34402[i];
    if e38416i5[i]=5 and e38416ib5[i]>0 and e34402[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])/(e34402[i]*4.3);
  /* missing value */
    if e38416i5[i] in (3,4,5) and -4<e34402[i]<=0 then othpay5[i]=-3;
   end;
```

```
  if e384075[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e34402[i]=0 then othpay5[i]=-3;
  if e384075[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e34402[i]<0 then
          othpay5[i]=e34402[i];

  end;
  end;

/*** case vi. more then one compensation at the beginning, diff no. of hours. ***/

if e38330[i]=1 and e20700[i]=1 and (e38102[i]=1 or e38102[i]=3) then do;

  /* for tips*/

  if e38103[i]>0 and e384071[i]=1 and e384161[i]>=0 then othpay1[i]=e384161[i];
  if e38103[i]>0 and e384071[i]=2 and e38416k1[i]>0 and e384161[i]>=0 then
          othpay1[i]=(e384161[i]*e38416k1[i])/e38103[i];
  if e38103[i]>0 and e384071[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384161[i]>=0
          then othpay1[i]=e384161[i]/e38103[i];
  if e38103[i]>0 and e384071[i]=4 and e384161[i]>=0 then othpay1[i]=e384161[i]/(2*e38103[i]);
  if e38103[i]>0 and e384071[i]=5 and e384161[i]>=0 then othpay1[i]=e384161[i]/(4.3*e38103[i]);
  if e38103[i]>0 and e384071[i]=6 and e384161[i]>=0 and e35600[i]>0 then
          othpay1[i]=e384161[i]/(e35600[i]*e38103[i]);
  if e38103[i]>0 and e384071[i]=8 and e384161[i]>=0 then othpay1[i]=e384161[i]/(2.15*e38103[i]);
  if e384071[i] in (9, 14) then othpay1[i]=0;
  if e384071[i] in (12,13) and e384161[i]>=o then do;
     if e38416g1[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
       if e225301[i] in (1,2,4,5) and e225001[i]>0 and othpay1[i]>=0 then
          othpay1[i]=othpay1[i]/e225001[i]*e384161[i];
       if e225301[i]=3 and e225001[i]>0 and e38416k1[i]>0 and e225501[i]>0 and othpay1[i]>=0
         then othpay1[i]=othpay1[i]/(e225001[i]*e225501[i])*e384161[i]*e38416k1[i];
       if e225301[i]=3 and e38416k1[i]<=0 then othpay1[i]=-3;
     end;
     else do; /*in all the other cases */
      if e38416i1[i]=1 and e38416ib1[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])*60;
      if e38416i1[i]=2 and e38416ib1[i]>0 then othpay1[i]=e384161[i]/e38416ib1[i];
      if e38416i1[i]=3 and e38416ib1[i]>0 and e38416k1[i]>=0 and e38103[i]>0 then
        othpay1[i]=(e384161[i]/e38416ib1[i])*e38416k1[i]/e38103[i];
      if e38416i1[i]=4 and e38416ib1[i]>0 and e38103[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])/e38103[i];
      if e38416i1[i]=5 and e38416ib1[i]>0 and e38103[i]>0 then othpay1[i]=(e384161[i]/e38416ib1[i])/(e38103[i]*4.3);
    /* missing value */
      if -4<e38416i1[i]<0 or -4<e38416ib1[i]<=0 then othpay1[i]=-3;
      if e38416i1[i] in (3,4,5) and -4<e38103[i]<=0 then othpay1[i]=-3;
      if e38416i1[i]=3 and e38416k1[i]<0 then othpay1[i]=-3;
     end;
   end;

  if e384071[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384161[i]<0 then
          othpay1[i]=e225001[i];
  if e384071[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e38103[i]=0 then othpay1[i]=-3;
  if e384071[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38103[i]<0 then
          othpay1[i]=e38103[i];
  if e384071[i]=2 and e38416k1[i] le 0 then othpay1[i]=-3;
  if e384071[i]=2 and -4<e38416k1[i]<0 then othpay1[i]=e225501[i];
  if e384071[i]=6 and e35600[i] le 0 then othpay1[i]=-3;
  if e384071[i]=6 and -4<e35600[i]<0 then othpay1[i]=e35600[i];
```

```
/*for commissions*/

  if e38103[i]>0 and e384072[i]=1 and e384162[i]>=0 then othpay2[i]=e384162[i];
  if e38103[i]>0 and e384072[i]=2 and e38416k2[i]>0 and e384162[i]>=0 then
          othpay2[i]=(e384162[i]*e38416k2[i])/e38103[i];
  if e38103[i]>0 and e384072[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384162[i]>=0
          then othpay2[i]=e384162[i]/e38103[i];
  if e38103[i]>0 and e384072[i]=4 and e384162[i]>=0 then othpay2[i]=e384162[i]/(2*e38103[i]);
  if e38103[i]>0 and e384072[i]=5 and e384162[i]>=0 then othpay2[i]=e384162[i]/(4.3*e38103[i]);
  if e38103[i]>0 and e384072[i]=6 and e384162[i]>=0 and e35600[i]>0 then
          othpay2[i]=e384162[i]/(e35600[i]*e38103[i]);
  if e38103[i]>0 and e384072[i]=8 and e384162[i]>=0 then othpay2[i]=e384162[i]/(2.15*e38103[i]);
  if e384071[i] in (9, 14) then othpay1[i]=0;
  if e384072[i] in (12,13) and e384162[i]>=o then do;
     if e38416g2[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
       if e225302[i] in (1,2,4,5) and e225002[i]>0 and othpay2[i]>=0 then
          othpay2[i]=othpay2[i]/e225002[i]*e384162[i];
       if e225302[i]=3 and e225002[i]>0 and e38416k2[i]>0 and e225502[i]>0 and othpay2[i]>=0
         then othpay2[i]=othpay2[i]/(e225002[i]*e225502[i])*e384162[i]*e38416k2[i];
       if e225302[i]=3 and e38416k2[i]<=0 then othpay2[i]=-3;
     end;
     else do; /*in all the other cases */
      if e38416i2[i]=1 and e38416ib2[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])*60;
      if e38416i2[i]=2 and e38416ib2[i]>0 then othpay2[i]=e384162[i]/e38416ib2[i];
      if e38416i2[i]=3 and e38416ib2[i]>0 and e38416k2[i]>=0 and e38103[i]>0 then
        othpay2[i]=(e384162[i]/e38416ib2[i])*e38416k2[i]/e38103[i];
      if e38416i2[i]=4 and e38416ib2[i]>0 and e38103[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])/e38103[i];
      if e38416i2[i]=5 and e38416ib2[i]>0 and e38103[i]>0 then othpay2[i]=(e384162[i]/e38416ib2[i])/(e38103[i]*4.3);
    /* missing value */
      if -4<e38416i2[i]<0 or -4<e38416ib2[i]<=0 then othpay2[i]=-3;
      if e38416i2[i] in (3,4,5) and -4<e38103[i]<=0 then othpay2[i]=-3;
      if e38416i2[i]=3 and e38416k2[i]<0 then othpay2[i]=-3;
     end;
   end;

  if e384072[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384162[i]<0 then
          othpay2[i]=e225002[i];
  if e384072[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e38103[i]=0 then othpay2[i]=-3;
  if e384072[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38103[i]<0 then
          othpay2[i]=e38103[i];
  if e384072[i]=2 and e38416k2[i] le 0 then othpay2[i]=-3;
  if e384072[i]=2 and -4<e38416k2[i]<0 then othpay2[i]=e225502[i];
  if e384072[i]=6 and e35600[i] le 0 then othpay2[i]=-3;
  if e384072[i]=6 and -4<e35600[i]<0 then othpay2[i]=e35600[i];

/*for bonuses*/

  if e38103[i]>0 and e384073[i]=1 and e384163[i]>=0 then othpay3[i]=e384163[i];
  if e38103[i]>0 and e384073[i]=2 and e38416k3[i]>0 and e384163[i]>=0 then
          othpay3[i]=(e384163[i]*e38416k3[i])/e38103[i];
  if e38103[i]>0 and e384073[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384163[i]>=0
          then othpay3[i]=e384163[i]/e38103[i];
  if e38103[i]>0 and e384073[i]=4 and e384163[i]>=0 then othpay3[i]=e384163[i]/(2*e38103[i]);
  if e38103[i]>0 and e384073[i]=5 and e384163[i]>=0 then othpay3[i]=e384163[i]/(4.3*e38103[i]);
  if e38103[i]>0 and e384073[i]=6 and e384163[i]>=0 and e35600[i]>0 then
          othpay3[i]=e384163[i]/(e35600[i]*e38103[i]);
  if e38103[i]>0 and e384073[i]=8 and e384163[i]>=0 then othpay3[i]=e384163[i]/(2.15*e38103[i]);
```

```
 if e384073[i] in (9, 14) then othpay3[i]=0;
 if e384073[i] in (12,13) and e384163[i]>=o then do;
    if e38416g3[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
      if e225303[i] in (1,2,4,5) and e225003[i]>0 and othpay3[i]>=0 then
          othpay3[i]=othpay3[i]/e225003[i]*e384163[i];
      if e225303[i]=3 and e225003[i]>0 and e38416k3[i]>0 and e225503[i]>0 and othpay3[i]>=0
         then othpay3[i]=othpay3[i]/(e225003[i]*e225503[i])*e384163[i]*e38416k3[i];
      if e225303[i]=3 and e38416k3[i]<=0 then othpay3[i]=-3;
    end;
    else do; /*in all the other cases */
     if e38416i3[i]=1 and e38416ib3[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])*60;
     if e38416i3[i]=2 and e38416ib3[i]>0 then othpay3[i]=e384163[i]/e38416ib3[i];
     if e38416i3[i]=3 and e38416ib3[i]>0 and e38416k3[i]>=0 and e38103[i]>0 then
       othpay3[i]=(e384163[i]/e38416ib3[i])*e38416k3[i]/e38103[i];
     if e38416i3[i]=4 and e38416ib3[i]>0 and e38103[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])/e38103[i];
     if e38416i3[i]=5 and e38416ib3[i]>0 and e38103[i]>0 then othpay3[i]=(e384163[i]/e38416ib3[i])/(e38103[i]*4.3);
   /* missing value */
     if -4<e38416i3[i]<0 or -4<e38416ib3[i]<=0 then othpay3[i]=-3;
     if e38416i3[i] in (3,4,5) and -4<e38103[i]<=0 then othpay3[i]=-3;
     if e38416i3[i]=3 and e38416k3[i]<0 then othpay3[i]=-3;
    end;
  end;

 if e384073[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384163[i]<0 then
          othpay3[i]=e225003[i];
 if e384073[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e38103[i]=0 then othpay3[i]=-3;
 if e384073[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38103[i]<0 then
          othpay3[i]=e38103[i];
 if e384073[i]=2 and e38416k3[i] le 0 then othpay3[i]=-3;
 if e384073[i]=2 and -4<e38416k3[i]<0 then othpay3[i]=e225503[i];
 if e384073[i]=6 and e35600[i] le 0 then othpay3[i]=-3;
 if e384073[i]=6 and -4<e35600[i]<0 then othpay3[i]=e35600[i];

/*for incentive pay*/

 if e38103[i]>0 and e384074[i]=1 and e384164[i]>=0 then othpay4[i]=e384164[i];
 if e38103[i]>0 and e384074[i]=2 and e38416k4[i]>0 and e384164[i]>=0 then
          othpay4[i]=(e384164[i]*e38416k4[i])/e38103[i];
 if e38103[i]>0 and e384074[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384164[i]>=0
          then othpay4[i]=e384164[i]/e38103[i];
 if e38103[i]>0 and e384074[i]=4 and e384164[i]>=0 then othpay4[i]=e384164[i]/(2*e38103[i]);
 if e38103[i]>0 and e384074[i]=5 and e384164[i]>=0 then othpay4[i]=e384164[i]/(4.3*e38103[i]);
 if e38103[i]>0 and e384074[i]=6 and e384164[i]>=0 and e35600[i]>0 then
          othpay4[i]=e384164[i]/(e35600[i]*e38103[i]);
 if e38103[i]>0 and e384074[i]=8 and e384164[i]>=0 then othpay4[i]=e384164[i]/(2.15*e38103[i]);
 if e384074[i] in (9, 14) then othpay4[i]=0;
 if e384074[i] in (12,13) and e384164[i]>=o then do;
    if e38416g4[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
      if e225304[i] in (1,2,4,5) and e225004[i]>0 and othpay4[i]>=0 then
          othpay4[i]=othpay4[i]/e225004[i]*e384164[i];
      if e225304[i]=3 and e225004[i]>0 and e38416k4[i]>0 and e225504[i]>0 and othpay4[i]>=0
         then othpay4[i]=othpay4[i]/(e225004[i]*e225504[i])*e384164[i]*e38416k4[i];
      if e225304[i]=3 and e38416k4[i]<=0 then othpay4[i]=-3;
    end;
    else do; /*in all the other cases */
     if e38416i4[i]=1 and e38416ib4[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])*60;
     if e38416i4[i]=2 and e38416ib4[i]>0 then othpay4[i]=e384164[i]/e38416ib4[i];
```

```
      if e38416i4[i]=3 and e38416ib4[i]>0 and e38416k4[i]>=0 and e38103[i]>0 then
        othpay4[i]=(e384164[i]/e38416ib4[i])*e38416k4[i]/e38103[i];
      if e38416i4[i]=4 and e38416ib4[i]>0 and e38103[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])/e38103[i];
      if e38416i4[i]=5 and e38416ib4[i]>0 and e38103[i]>0 then othpay4[i]=(e384164[i]/e38416ib4[i])/(e38103[i]*4.3);
    /* missing value */
      if -4<e38416i4[i]<0 or -4<e38416ib4[i]<=0 then othpay4[i]=-3;
      if e38416i4[i] in (3,4,5) and -4<e38103[i]<=0 then othpay4[i]=-3;
      if e38416i4[i]=3 and e38416k4[i]<0 then othpay4[i]=-3;
     end;
  end;

  if e384074[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384164[i]<0 then
          othpay4[i]=e225004[i];
  if e384074[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e38103[i]=0 then othpay4[i]=-3;
  if e384074[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38103[i]<0 then
          othpay4[i]=e38103[i];
  if e384074[i]=2 and e38416k4[i] le 0 then othpay4[i]=-3;
  if e384074[i]=2 and -4<e38416k4[i]<0 then othpay4[i]=e225504[i];
  if e384074[i]=6 and e35600[i] le 0 then othpay4[i]=-3;
  if e384074[i]=6 and -4<e35600[i]<0 then othpay4[i]=e35600[i];

/*for others*/

  if e38103[i]>0 and e384075[i]=1 and e384165[i]>=0 then othpay5[i]=e384165[i];
  if e38103[i]>0 and e384075[i]=2 and e38416k5[i]>0 and e384165[i]>=0 then
          othpay5[i]=(e384165[i]*e38416k5[i])/e38103[i];
  if e38103[i]>0 and e384075[i] in (3, 7,15, 16, 17, 18, 19, 21, 22, 23, 24, 25, 26, 28,999,-1,-2) and e384165[i]>=0
          then othpay5[i]=e384165[i]/e38103[i];
  if e38103[i]>0 and e384075[i]=4 and e384165[i]>=0 then othpay5[i]=e384165[i]/(2*e38103[i]);
  if e38103[i]>0 and e384075[i]=5 and e384165[i]>=0 then othpay5[i]=e384165[i]/(4.3*e38103[i]);
  if e38103[i]>0 and e384075[i]=6 and e384165[i]>=0 and e35600[i]>0 then
          othpay5[i]=e384165[i]/(e35600[i]*e38103[i]);
  if e38103[i]>0 and e384075[i]=8 and e384165[i]>=0 then othpay5[i]=e384165[i]/(2.15*e38103[i]);
  if e384075[i] in (9, 14) then othpay5[i]=0;
  if e384075[i] in (12,13) and e384165[i]>=o then do;
     if e38416g5[i]=1 then do; /* if the speed info. given at the beginning doesn't change. */
       if e225305[i] in (1,2,4,5) and e225005[i]>0 and othpay5[i]>=0 then
          othpay5[i]=othpay5[i]/e225005[i]*e384165[i];
       if e225305[i]=3 and e225005[i]>0 and e38416k5[i]>0 and e225505[i]>0 and othpay5[i]>=0
         then othpay5[i]=othpay5[i]/(e225005[i]*e225505[i])*e384165[i]*e38416k5[i];
       if e225305[i]=3 and e38416k5[i]<=0 then othpay5[i]=-3;
     end;
     else do; /*in all the other cases */
      if e38416i5[i]=1 and e38416ib5[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])*60;
      if e38416i5[i]=2 and e38416ib5[i]>0 then othpay5[i]=e384165[i]/e38416ib5[i];
      if e38416i5[i]=3 and e38416ib5[i]>0 and e38416k5[i]>=0 and e38103[i]>0 then
        othpay5[i]=(e384165[i]/e38416ib5[i])*e38416k5[i]/e38103[i];
      if e38416i5[i]=4 and e38416ib5[i]>0 and e38103[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])/e38103[i];
      if e38416i5[i]=5 and e38416ib5[i]>0 and e38103[i]>0 then othpay5[i]=(e384165[i]/e38416ib5[i])/(e38103[i]*4.3);
    /* missing value */
      if -4<e38416i5[i]<0 or -4<e38416ib5[i]<=0 then othpay5[i]=-3;
      if e38416i5[i] in (3,4,5) and -4<e38103[i]<=0 then othpay5[i]=-3;
      if e38416i5[i]=3 and e38416k5[i]<0 then othpay5[i]=-3;
     end;
  end;
```

```
  if e384075[i] in (1,2,3,4,5,6,7,8,12,13,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e384165[i]<0 then
        othpay5[i]=e225005[i];
  if e384075[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and e38103[i]=0 then othpay5[i]=-3;
  if e384075[i] in (2,3,4,5,6,7,8,15,16,17,18,19,21,22,23,24,25,26,28,999,-1,-2) and -4<e38103[i]<0 then
        othpay5[i]=e38103[i];
  if e384075[i]=2 and e38416k5[i] le 0 then othpay5[i]=-3;
  if e384075[i]=2 and -4<e38416k5[i]<0 then othpay5[i]=e225505[i];
  if e384075[i]=6 and e35600[i] le 0 then othpay5[i]=-3;
  if e384075[i]=6 and -4<e35600[i]<0 then othpay5[i]=e35600[i];
  end;
end;
end;

/******************* overall end hourly compensation ****************************************/

do i=1 to 9;
 if e37901b[i]=1 or e59900[i]=1 then do;
  hrcomp[i]=0;
  if hrwg[i] ge 0 then hrcomp[i]=hrcomp[i]+hrwg[i];
  if ot[i] ge 0 then hrcomp[i]=hrcomp[i]+ot[i];
  if othpay1[i] ge 0 then hrcomp[i]=hrcomp[i]+othpay1[i];
  if othpay2[i] ge 0 then hrcomp[i]=hrcomp[i]+othpay2[i];
  if othpay3[i] ge 0 then hrcomp[i]=hrcomp[i]+othpay3[i];
  if othpay4[i] ge 0 then hrcomp[i]=hrcomp[i]+othpay4[i];
  if othpay5[i] ge 0 then hrcomp[i]=hrcomp[i]+othpay5[i];
  if -4<hrwg[i]<0 or -4<ot[i]<0 or -4<othpay1[i]<0 or -4<othpay2[i]<0 or -4<othpay3[i]<0 or -4<othpay4[i]<0 or -
        4<othpay5[i]<0 then hrcomp[i]=-3;
  if hrwg[i]=-4 then hrcomp[i]=-4;
 end;
end;

/* fix the hourly compensation because the respondents may understand the overtime question differently. 1) if overtime
        pay is less than or equal to the hourly wage, we assume the resp. consider the overtime as something above the
        hourly wage. so the hourly comp. should include both hourly wage and overtime pay.  2) if overtime pay is
        greater than the hourly wage, we assume the respondents consider the overtime pay as a pay including hourly
        wage. so the hourly comp. only include the overtime pay without the hourly wage. */

do i=1 to 9;
  if hrwg[i] gt 0 and ot[i] gt hrwg[i] and hrcomp[i] gt 0 then hrcomp[i]=hrcomp[i]-hrwg[i];
end;

/***                    fix round 4 hourly compensation calculation.  I will call the hourly wage, hourly overtime and hourly
        other compensation calculated before as previous computed hourly wage,etc.  To fix it, I will first find the
        overtime hours since I only need to fix those people who have overtime hours.  For the people who have
        overtime hours, the fixed total hourly comp. equals:
                ( (hrly wage)*(reg. hours) + (prev. hrly ot)*(ot hrs) + (prev. other hrly comp)*(reg. hrs) ) / total hrs.
        All the "hours" in the above equation refers to "weekly hours".
There are 2 exceptions to the above equation. When the time unit to other compensation is hrly or per item (and minute
        or hour as unit for each item), we multiply (prev. other hrly comp) by (total hrs) instead of (reg hrs). This term
        should give us the weekly other compensation. since previously we use regular hours to calculate hrly other
        compensation regardless, we use reg hrs in the equation above to recover the previous calculated hrly other
        compensation to the weekly other comepensation. but if the reported time unit is hrly, we don't have to recover
        anything and just use the total weekly hours. */

array reghr reghr1-reghr9;                array othr othr1-othr9;
array ttlhr ttlhr1-ttlhr9;                array pay1hr pay1hr1-pay1hr9;
array pay2hr pay2hr1-pay2hr9;             array pay3hr pay3hr1-pay3hr9;
```

```
array pay4hr pay4hr1-pay4hr9;                array pay5hr pay5hr1-pay5hr9;
array newcomp newcomp1-newcomp9;
new=0;

array hrwgtp hrwgtp1-hrwgtp9;                array ottp ottp1-ottp9;
array pay1tp pay1tp1-pay1tp9;                array pay2tp pay2tp1-pay2tp9;
array pay3tp pay3tp1-pay3tp9;                array pay4tp pay4tp1-pay4tp9;
array pay5tp pay5tp1-pay5tp9;                array odreghr odreghr1-odreghr9;

do i=1 to 9; reghr[i]=0; othr[i]=0; end;

***overtime hours;

do i=1 to 9; /* loop 1. */

 if e19200[i] not in (-4,-5) then do;
   if e19200[i]=1 and e24501[i] not in (., -4) then othr[i]=e24501[i];
   if e19200[i] ne 1 and e34403[i] not in (., -4) then othr[i]=e34403[i];
   if e22611[i] not in (., -4) then othr[i]=e22611[i];
 end;
 if e83100[i] not in (-4,-5) then do;
   if e83100[i]=1 and e88501[i] not in (., -4) then othr[i]=e88501[i];
   if e83100[i] ne 1 and e98403[i] not in (., -4) then othr[i]=e98403[i];
   if e100232[i] not in (., -4) then othr[i]=e100232[i];
 end;
 if e37901b[i]=1 or e59900[i]=1 then do;
   if e38002[i] not in (., -4) then othr[i]=e38002[i];
   if e38105[i] not in (., -4) then othr[i]=e38105[i];
 end;

if othr[i]=0 then newcomp[i]=hrcomp[i];
else if othr[i]<0 then newcomp[i]=-3;
else if othr[i]>0 then do; /*loop 2. */
  new=1;

  ***get the regular hours;
  if e19200[i] not in (-4,-5) then do;
    if e19200[i]=1 and e23901[i] not in (., -4) then reghr[i]=e23901[i];
    if e19200[i] ne 1 and e34402[i] not in (., -4) then reghr[i]=e34402[i];
    if e19200[i] ne 1 and e34428[i] not in (., -4) then reghr[i]=e34428[i];
    if e22610[i] not in (-4,0) then reghr[i]=e22610[i];
  end;
  if e83100[i] not in (-4,-5) then do;
    if e83100[i]=1 and e88000[i] not in (., -4) then reghr[i]=e88000[i];
    if e83100[i] ne 1 and e98402[i] not in (., -4) then reghr[i]=e98402[i];
    if e83100[i] ne 1 and e98429[i] not in (., -4) then reghr[i]=e98429[i];
    if e100231[i] not in (., -4) then reghr[i]=e100231[i];
  end;
  if e37901b[i]=1 or e59900[i]=1 then do;
    if e3800ff[i] not in (., -4) then reghr[i]=e3800ff[i];
    if e38103[i] not in (., -4) then reghr[i]=e38103[i];
  end;

  if reghr[i]<0 then newcomp[i]=-3;
  else if reghr[i] ge 0 then do; /*loop 3. */
    ttlhr[i]=reghr[i]+othr[i];
```

```
  ***decide what hours (reg. hours or total hours or old reg. hours) to multiply by previous calculated hourly
  ***compensation depending on the time unit for the compensation;

  pay1hr[i]=reghr[i]; pay2hr[i]=reghr[i]; pay3hr[i]=reghr[i]; pay4hr[i]=reghr[i]; pay5hr[i]=reghr[i];

 if e19200[i] not in (-4,-5) then do;
   if e19200[i]=1 and e23901[i] ne -4 then odreghr[i]=e23901[i];
   if e19200[i] ne 1 and e34402[i] ne -4 then odreghr[i]=e34402[i];
   if e19200[i] ne 1 and e34428[i] ne -4 then odreghr[i]=e34428[i];
   if e22610[i] ne -4 then odreghr[i]=e22610[i];
 end;
 if e83100[i] not in (-4,-5) then do;
   if e83100[i]=1 and e88000[i] ne -4 then odreghr[i]=e88000[i];
   if e83100[i] ne 1 and e98402[i] ne -4 then odreghr[i]=e98402[i];
   if e83100[i] ne 1 and e98429[i] ne -4 then odreghr[i]=e98429[i];
   if e100231[i] ne -4 then odreghr[i]=e100231[i];
 end;

if (e38103[i] ne -4 or e3800ff[i] ne -4) then do;
   if othpay1[i] ge 0 and (e38329b[i]=-4 and e384071[i]=-4) then pay1hr[i]=odreghr[i];
    /* this is when we didn't fix comp. according to the new hours. */
   if othpay2[i] ge 0 and (e38329b[i]=-4 and e384072[i]=-4) then pay2hr[i]=odreghr[i];
   if othpay3[i] ge 0 and (e38329b[i]=-4 and e384073[i]=-4) then pay3hr[i]=odreghr[i];
   if othpay4[i] ge 0 and (e38329b[i]=-4 and e384074[i]=-4) then pay4hr[i]=odreghr[i];
   if othpay5[i] ge 0 and (e38329b[i]=-4 and e384075[i]=-4) then pay5hr[i]=odreghr[i];
end;

   if e19200[i] not in (-4,-5) and e20700[i]=1 then do;
     if e216001[i]=1 or ( e216001[i] in (12,13) and e225301[i] in (1,2)) then pay1hr[i]=ttlhr[i];
     if e216002[i]=1 or ( e216002[i] in (12,13) and e225302[i] in (1,2)) then pay2hr[i]=ttlhr[i];
     if e216003[i]=1 or ( e216003[i] in (12,13) and e225303[i] in (1,2)) then pay3hr[i]=ttlhr[i];
     if e216004[i]=1 or ( e216004[i] in (12,13) and e225304[i] in (1,2)) then pay4hr[i]=ttlhr[i];
     if e216005[i]=1 or ( e216005[i] in (12,13) and e225305[i] in (1,2)) then pay5hr[i]=ttlhr[i];
     if e22613[i]=1 or ( e22613[i] in (12,13) and e22628c[i] in (1,2)) then pay1hr[i]=ttlhr[i];
     if e22614[i]=1 or ( e22614[i] in (12,13) and e22629c[i] in (1,2)) then pay2hr[i]=ttlhr[i];
     if e22615[i]=1 or ( e22615[i] in (12,13) and e22630c[i] in (1,2)) then pay3hr[i]=ttlhr[i];
     if e22616[i]=1 or ( e22616[i] in (12,13) and e22631c[i] in (1,2)) then pay4hr[i]=ttlhr[i];
     if e22617[i]=1 or ( e22617[i] in (12,13) and e22632c[i] in (1,2)) then pay5hr[i]=ttlhr[i];
   end;
   if e83100[i] not in (-4,-5) then do;
     if e102051[i]=1 or ( e102051[i] in (12,13) and e10214aaa1[i] in (1,2)) then pay1hr[i]=ttlhr[i];
     if e102052[i]=1 or ( e102052[i] in (12,13) and e10214aaa2[i] in (1,2)) then pay2hr[i]=ttlhr[i];
     if e102053[i]=1 or ( e102053[i] in (12,13) and e10214aaa3[i] in (1,2)) then pay3hr[i]=ttlhr[i];
     if e102054[i]=1 or ( e102054[i] in (12,13) and e10214aaa4[i] in (1,2)) then pay4hr[i]=ttlhr[i];
     if e102055[i]=1 or ( e102055[i] in (12,13) and e10214aaa5[i] in (1,2)) then pay5hr[i]=ttlhr[i];
     if e100234[i]=1 or ( e100234[i] in (12,13) and e100250c[i] in (1,2)) then pay1hr[i]=ttlhr[i];
     if e100235[i]=1 or ( e100235[i] in (12,13) and e100251c[i] in (1,2)) then pay2hr[i]=ttlhr[i];
     if e100236[i]=1 or ( e100236[i] in (12,13) and e100252c[i] in (1,2)) then pay3hr[i]=ttlhr[i];
     if e100237[i]=1 or ( e100237[i] in (12,13) and e100253c[i] in (1,2)) then pay4hr[i]=ttlhr[i];
     if e100239[i]=1 or ( e100239[i] in (12,13) and e100254c[i] in (1,2)) thne pay5hr[i]=ttlhr[i];
   end;
   if e37901b[i]=1 or e59900[i]=1 then do;
     if e384071[i]=1 or (e384071[i] in (12,13) and e38416g1[i]=1 and e225301[i] in (1,2))
        or (e384071[i] in (12,13) and e38416i1[i] in (1,2)) then pay1hr[i]=ttlhr[i];
     if e384072[i]=1 or (e384072[i] in (12,13) and e38416g2[i]=1 and e225302[i] in (1,2))
        or (e384072[i] in (12,13) and e38416i2[i] in (1,2)) then pay2hr[i]=ttlhr[i];
     if e384073[i]=1 or (e384073[i] in (12,13) and e38416g3[i]=1 and e225303[i] in (1,2))
```

```
      or (e384073[i] in (12,13) and e38416i3[i] in (1,2)) then pay3hr[i]=ttlhr[i];
    if e384074[i]=1 or (e384074[i] in (12,13) and e38416g4[i]=1 and e225304[i] in (1,2))
      or (e384074[i] in (12,13) and e38416i4[i] in (1,2)) then pay4hr[i]=ttlhr[i];
    if e384075[i]=1 or (e384075[i] in (12,13) and e38416g5[i]=1 and e225305[i] in (1,2))
      or (e384075[i] in (12,13) and e38416i5[i] in (1,2)) then pay5hr[i]=ttlhr[i];
    if e38329[i]=1 and e20700[i]=1 then do;
      if e212002[i]=1 and (e38329b[i]=1 or ( e38329b[i] in (12,13) and e38329g[i]=1 and e225301[i] in (1,2))
        or (e38329b[i] in (12,13) and e38329i[i] in (1,2)) ) then pay1hr[i]=ttlhr[i];
      if e212003[i]=1 and (e38329b[i]=1 or ( e38329b[i] in (12,13) and e38329g[i]=1 and e225302[i] in (1,2))
        or (e38329b[i] in (12,13) and e38329i[i] in (1,2)) ) then pay2hr[i]=ttlhr[i];
      if e212004[i]=1 and (e38329b[i]=1 or ( e38329b[i] in (12,13) and e38329g[i]=1 and e225303[i] in (1,2))
        or (e38329b[i] in (12,13) and e38329i[i] in (1,2)) ) then pay3hr[i]=ttlhr[i];
      if e212005[i]=1 and (e38329b[i]=1 or ( e38329b[i] in (12,13) and e38329g[i]=1 and e225304[i] in (1,2))
        or (e38329b[i] in (12,13) and e38329i[i] in (1,2)) ) then pay4hr[i]=ttlhr[i];
      if e212006[i]=1 and (e38329b[i]=1 or ( e38329b[i] in (12,13) and e38329g[i]=1 and e225305[i] in (1,2))
        or (e38329b[i] in (12,13) and e38329i[i] in (1,2)) ) then pay5hr[i]=ttlhr[i];
     end;
   end;

   ***finally, calculate the corrected total compensation;
    hrwgtp[i]=hrwg[i]; if hrwg[i]=-4 then hrwgtp[i]=0;
    ottp[i]=ot[i]; if ot[i]=-4 then ottp[i]=0;
    pay1tp[i]=othpay1[i]; if othpay1[i]=-4 then pay1tp[i]=0;
    pay2tp[i]=othpay2[i]; if othpay2[i]=-4 then pay2tp[i]=0;
    pay3tp[i]=othpay3[i]; if othpay3[i]=-4 then pay3tp[i]=0;
    pay4tp[i]=othpay4[i]; if othpay4[i]=-4 then pay4tp[i]=0;
    pay5tp[i]=othpay5[i]; if othpay5[i]=-4 then pay5tp[i]=0;

   if -4<hrwg[i]<0 or -4<ot[i]<0 or -4<othpay1[i]<0 or -4<othpay2[i]<0 or -4<othpay3[i]<0
      or -4<othpay4[i]<0 or -4<othpay5[i]<0 then newcomp[i]=-3;
   else newcomp[i]=(hrwgtp[i]*reghr[i]+ottp[i]*othr[i]+pay1tp[i]*pay1hr[i]+pay2tp[i]*pay2hr[i]+pay3tp[i]*pay3hr[i]
          +pay4tp[i]*pay4hr[i]+pay5tp[i]*pay5hr[i])/ttlhr[i];

  end; /* end of loop 3. */

 end; /* end of loop 2. */

end; /* end of loop 1. */

/********************** job lasts 13 weeks or less *******************************************/
jleng01=-4;      jleng02=-4;      jleng03=-4;      jleng04=-4;      jleng05=-4;
jleng06=-4;      jleng07=-4;      jleng08=-4;      jleng09=-4;
array jleng jleng01 jleng02 jleng03 jleng04 jleng05 jleng06 jleng07 jleng08 jleng09 jleng10;

do i=1 to 9;
  if e37701[i]=0 then jleng[i]=0;
  if e58201[i]=0 then jleng[i]=0;
  if e37901b[i]>-1 or e59900[i]>-1 or e58401[i]>-1 then do;
          if e37901b[i]=1 or e59900[i]=1 or e58401[i]=0 then jleng[i]=1;
          else jleng[i]=0;
  end;
end;

/*****************missing values*****************************/
array jleng jleng01 jleng02 jleng03 jleng04 jleng05 jleng06 jleng07 jleng08 jleng09;
array hrwg hrwg01 hrwg02 hrwg03 hrwg04 hrwg05 hrwg06 hrwg07 hrwg08 hrwg09 ;
array newcomp newcomp1-newcomp9;
```

```
do i=1 to 9;
   if jleng[i]>-4 and hrwg[i]=-4 then hrwg[i]=-3;
   if jleng[i]>-4 and newcomp[i]=-4 then newcomp[i]=-3;
end;

array hrwgr hrwgr01 hrwgr02 hrwgr03 hrwgr04 hrwgr05 hrwgr06 hrwgr07 hrwgr08 hrwgr09;
hrwgr01=-4;     hrwgr02=-4;     hrwgr03=-4;     hrwgr04=-4;     hrwgr05=-4;
hrwgr06=-4;     hrwgr07=-4;     hrwgr08=-4;     hrwgr09=-4;

do i=1 to 9;
  hrwgr[i]=round(hrwg[i],1);
  if e200a=-5 then hrwgr[i]=-5;
  if e200a=-5 then jleng[i]=-5;
end;

array hrcomr hrcomr01 hrcomr02 hrcomr03 hrcomr04 hrcomr05 hrcomr06 hrcomr07 hrcomr08 hrcomr09;
hrcomr01=-4;    hrcomr02=-4;    hrcomr03=-4;    hrcomr04=-4;    hrcomr05=-4;
hrcomr06=-4;    hrcomr07=-4;    hrcomr08=-4;    hrcomr09=-4;

do i=1 to 9;
  hrcomr[i]=round(newcomp[i],1);
  if e200a=-5 then hrcomr[i]=-5;
end;

/*hand edit*/ if id=1280802 then do; hrcomr02=-3; hrwgr02=-3; end;

endsas;
```

NLSY97 Appendix 3: Family Background and Formation Variable Creation

HOUSEHOLD SIZE AS OF THE SURVEY DATE

Variables Created: CV_HH_SIZE
CV_HH_UNDER_6
CV_HH_UNDER_18

Variables Used

Name in Program	Question Name on CD
hage01-hage14	HHI_AGE.01–.14
huid01-huid14	HHI_UID.01–.14
id	PUBID
rsage	SYMBOL!KEY!AGE

This program creates several variables describing the composition of the respondent's household: the total number of residents, the number of residents under age 6, and the number of residents under age 18.

```
array hage hage01-hage14;
array huid huid01-huid14;
array hhdum hhdum01-hhdum14;
array dum6 dum601-dum614;
array dum18 dum1801-dum1814;

do i=1 to 14;
  hhdum[i]=0; dum6[i]=0; dum18[i]=0;
end;

do i=1 to 14;
  if huid[i]>-1 then hhdum[i]=1;
  if -1<hage[i]<6 then dum6[i]=1;
  if -1<hage[i]<18 then dum18[i]=1;
end;

/* create the dummy for the respondent's age rdum6
      and rdum18. */
rdum6=0;
rdum18=0;

if -1<rsage<6 then rdum6=1;
if -1<rsage<18 then rdum18=1;

/* Create household size hhsize by adding up the
      dummies hhdum[i] and also add one due to the
      respondent himself.  Similarly, create variables
      under6 and under18 by adding up the other two
      dummies.*/
hhsiz=1;
under6=rdum6;
under18=rdum18;

do i=1 to 14;
  hhsiz=hhsiz+hhdum[i];
  under6=under6+dum6[i];
  under18=under18+dum18[i];
end;

if huid01=-5 then hhsiz=-5;
if -4<huid01<0 or -4<huid02<0 or -4<huid03<0 or
      -4<huid04<0 or -4<huid05<0 or -4<huid06<0 or
      -4<huid07<0 or -4<huid08<0 or -4<huid09<0 or
      -4<huid10<0 or -4<huid11<0 or -4<huid12<0 or
      -4<huid13<0 or -4<huid14<0 then hhsiz=-3;

if huid01=-5 then under6=-5;
if huid01=-5 then under18=-5;
if -4<hage01<0 or -4<hage02<0 or -4<hage03<0 or
      -4<hage04<0 or -4<hage05<0 or -4<hage06<0 or
      -4<hage07<0 or -4<hage08<0 or -4<hage09<0 or
      -4<hage10<0 or -4<hage11<0 or -4<hage12<0 or
      -4<hage13<0 or -4<hage14<0 or -4<rsage<0 then
      do;
  under6=-3;
  under18=-3;
end;

/* this respondent has household uid02 eq to -3 but has
      hhage02 eq to 40. created variable is correct to
      avoid -3 */
if id = 1820961 then hhsiz = 8;

endsas;
```

YOUTH'S RELATIONSHIP TO HOUSEHOLD PARENT FIGURE(S)

Variables Created: CV_YTH_REL_HH_CURRENT

Variables Used

Name in Program	Question Name on CD	Name in Program	Question Name on CD
Round 1:		**Round 4:**	
pubid	PUBID	hhage01-14	HHI_AGE.01-.14
yid	YOUTH_ID.01	hh2uid01-14	HHI_UID.01-.14
sh931-sh935	SH-93.01-.05	marry_1-14	HHI_MARSTAT.01-.14
hh1uid01-hh1uid16	HHI2_UID.01-.16	rel_1-14	HHI_RELY.01-.14
		age	SYMBOL!KEY!AGE

Codes for Created Variable

-5 = Non-Interview
-3 = Invalid Skip
1 = Both biological parents
2 = Two parents, biological mother
3 = Two parents, biological father
4 = Biological mother only
5 = Biological father only
6 = Adoptive parent(s)
7 = Foster parent(s)
8 = No parents, grandparents
9 = No parents, other relatives
10 = Anything else

This program creates a variable identifying the youth's relationship to the primary adults in the household. In round 4 (as in rounds 2 and 3) there was no information collected on the legal guardian of the youth, so it is not possible to determine whether respondents are living with non-parent relatives because they are guardians or because the living situation is better (e.g., closer to school, no rent). For this reason, youths above the age 18 and above are considered independent and, if they are not living with an identified parent or parent-figure (legal guardian), are put into the anything else category. Youths below the age of 18 who are not living with an identified parent or parent-figure are put into the category that most closely matches their household situation.

```
array rel_a (i) rel_1-rel_14;                array marry_a (i) marry_1-marry_14;
array gend_a (i) gend_1-gend_14;             array age_a (i) hhage01-hhage14;

*this part determines legal guardians so that we can determine whether any guardians are present from the R1 interview
        use only unique IDs (cases with 6 digits are hh members after R1 only, so this program won't do anything
        with them, as we don't have info on legal guardian status for these HH members ;
array hh1uid (i) hh1uid01-hh1uid14;          *round 1 variable;
array hh2uid (i) hh2uid01-hh2uid14;          *round 4 variable;

do i=1 to 14;
  if yid=1 and sh931>0 and sh931+100=hh1uid then do; uid=hh1uid; end;
  if yid=2 and sh932>0 and sh932+100=hh1uid then do; uid=hh1uid; end;
  if yid=3 and sh933>0 and sh933+100=hh1uid then do; uid=hh1uid; end;
  if yid=4 and sh934>0 and sh934+100=hh1uid then do; uid=hh1uid; end;
  if yid=5 and sh935>0 and sh935+100=hh1uid then do; uid=hh1uid; end;
end;
do i=1 to 14; if (uid ne . and uid=hh2uid) then line=i; end;

legal=0;
do i=1 to 14; if line=i then legal=rel_a; end;
momid=0;     domid=0;     adopdad=0;     admom=0;     fostma=0;     fostda=0;
stepma=0;    stepda=0;    husb=0;        wife=0;      grand=0;      relat=0;
nonrel=0;    indep=0;     spouse=0;

**legal;
do i=1 to 14;
    if (legal>28 and legal<37) then do; grand=1; end;
```

```
    *spouse;                  if legal=1 or legal=2 then do; spouse=1; end;
    *brother/sister;          if (legal>12 and legal<19) and age_a>20 then do; relat=1; end;
    *aunt/uncle/other rel.;   if (legal>69 and legal<85) and age_a>20 then do; relat=1; end;
    *lover, roommate, other non-relative, mom's or dad's partner;
    if legal=69 or legal=68 or legal=85 or legal=88 or legal=89 then do; nonrel=1; end;
    if legal=-1 or legal=-2 or legal=-3 then do; invalid=1; end;
    if rel_1=-4 and rel_2=-4 and rel_3=-4 and rel_4=-4 and rel_5=-4 and rel_6=-4 and rel_7=-4 and rel_8=-4 and
          rel_9=-4 and rel_10=-4 and rel_11=-4 and rel_12=-4 and rel_13=-4 and rel_14=-4 then do;
       indep=1;
    end;
end;

**not legal;
do i=1 to 14;
    if (rel_a>28 and rel_a<37) then do; nlgrand=1; end;
    *spouse;                  if rel_a=1 or rel_a=2 then do; nlspouse=1; end;
    *brother/sister;          if (rel_a>12 and rel_a<19) and age_a>20 then do; nlrelat=1; end;
    *aunt/uncle/other rel.;   if (rel_a>69 and rel_a<85) and age_a>20 then do; nlrelat=1; end;
    *lover, roommate, other non-relative, mom's or dad's partner;
    if rel_a=69 or rel_a=68 or rel_a=85 or rel_a=88 or rel_a=89 then do; nlnnrl=1; end;
    if rel_a=-1 or rel_a=-2 or rel_a=-3 then do; nlinv=1; end;
  end;

/*problem is a variable to check whether R did not report a relationship code for one or more household members*/
problem=0;
do i=1 to 14; if rel_a=-1 or rel_a=-2 or rel_a=-3 then problem=1; end;

if age<18 then do;
   if nlgrand ne . and legal=0 then grand=nlgrand;          if nlspouse ne . and legal=0 then spouse=nlspouse;
   if nlrelat ne . and legal=0 then relat=nlrelat;          if nlnnrl ne . and legal=0 then nonrel=nlnnrl;
   if nlinv ne . and legal=0 then invalid=nlinv;
end;

*for all youths;
if nlspouse ne . then spouse=nlspouse;
do i=1 to 14;
    if rel_a=3 then momid=i;          if rel_a=4 then domid=i;
    if rel_a=5 then admom=i;          if rel_a=6 then adopdad=i;
    if rel_a=7 then stepma=i;         if rel_a=8 then stepda=i;
    if rel_a=9 then fostma=i;         if rel_a=10 then fostda=i;
 end;

rel=-16;
if age>17 then do; rel=10; end;                        if marry_1=-5 then do; rel=-5; end;
if indep=1 then do; rel=10; end;                       if invalid=1 then do; rel=-3; end;
if nonrel>0 then do; rel=10; end;                      if relat>0 then do; rel=9; end;
if grand>0 and momid=0 and domid=0 then do; rel=8; end;
if spouse=1 then do; rel=10; end;                      if fostda>0 or fostma>0 then do; rel=7; end;
if admom>0 or adopdad>0 then do; rel=6; end;           if stepda>0 or stepma>0 then do; rel=10; end;
if domid>0 and momid=0 then do; rel=5; end;            if momid>0 and domid=0 then do; rel=4; end;
if domid>0 and momid=0 then do; if admom>0 or stepma>0 then rel=3; end;
if momid>0 and domid=0 then do; if adopdad>0 or stepda>0 then rel=2; end;
if momid>0 and domid>0 then do; both=1; rel=1; end;

if rel=-16 then rel=10;

endsas;
```

YOUTH'S MARITAL STATUS AND MARITAL/COHABITATION HISTORY

Variables Created:

CV_MARSTAT	CV_MARSTAT_COLLAPSED
CV_FIRST_COHAB_DATE_M	CV_FIRST_COHAB_DATE_Y
CV_FIRST_MARRY_DATE_M	CV_FIRST_MARRY_DATE_Y
CV_FIRST_COHAB_MONTH	CV_FIRST_MARRY_MONTH
CV_COHAB_TTL	CV_MARRIAGES_TTL

Variables Used

Name in Program	Question Name on CD	Name in Program	Question Name on CD
id	PUBID	m560022m, m560022y	YMAR-5600.02.02~M, ~Y
int_d, int_m, int_y	YINTDATE~D, ~M, ~Y	m570011m, m570011y	YMAR-5700.01.01~M, ~Y
m620	YMAR-620	m570021m, m570021y	YMAR-5700.02.01~M, ~Y
m650	YMAR-650	m610011, m610012	YMAR-6100.01.01, .02
m700-m718	YMAR-700-YMAR-718	m610021, m610022	YMAR-6100.02.01, .02
m729d	YMAR-729D	m63001-m63003	YMAR-6300.01-.03
m730	YMAR-730	m64001-m64003	YMAR-6400.01-.03
m740	YMAR-740	m700011m, m700011y	YMAR-7000.01.01~M, ~Y
m760	YMAR-760	m700012m, m700012y	YMAR-7000.01.02~M, ~Y
m1000	YMAR-1000	m700013m, m700013y	YMAR-7000.01.03~M, ~Y
m1500	YMAR-1500	m720011-m720013	YMAR-7200.01.01-.03
m2600	YMAR-2600	m727011-m727041	YMAR-7270.01.01-.04.01
m2800	YMAR-2800	m730011, m730021	YMAR-7300.01.01, .02.01
m30501-m30504	YMAR-3050.01-.04	m735011-m735041	YMAR-7350.01.01-.04.01
m31001m, m31001y	YMAR-3100.01~M, ~Y	m7900111, m7900211	YMAR-7900.01.01.01, .02.01.01
m31002m, m31002y	YMAR-3100.02~M, ~Y	m810111m, m810111y	YMAR-8100.01.01.01~M, ~Y
m31003m, m31003y	YMAR-3100.03~M, ~Y	m820111m, m820111y	YMAR-8200.01.01.01~M, ~Y
m31004m, m31004y	YMAR-3100.04~M, ~Y	m820211m, m820211y	YMAR-8200.02.01.01~M, ~Y
m45001-m45004	YMAR-4500.01-.04	m910011-m910013	YMAR-9100.01.01-.03
m46501-m46504	YMAR-4650.01-.04	m920011m, m920011y	YMAR-9200.01.01~M, ~Y
m46701	YMAR-4670.01	m920012m, m920012y	YMAR-9200.01.02~M, ~Y
m47001, m47004	YMAR-4700.01-.04	m920013m, m920013y	YMAR-9200.01.03~M, ~Y
m48001m, m48001y	YMAR-4800.01~M, ~Y	m9800111	YMAR-9800.01.01.01
m48002m, m48002y	YMAR-4800.02~M, ~Y	m101111m, m101111y	YMAR-10100.01.01.01~M, ~Y
m48003m, m48003y	YMAR-4800.03~M, ~Y	m1020111	YMAR-10200.01.01.01
m49001-m49003	YMAR-4900.01-.03	m1080011-m1080013	YMAR-10800.01.01-.03
m540011, m540012	YMAR-5400.01.01, .02	m114001-m114003	YMAR-11400.01-.03
m540021, m540022	YMAR-5400.02.01, .02	m117001m, m117001y	YMAR-11700.01~M, ~Y
m560011m, m560011y	YMAR-5600.01.01~M, ~Y	m117002m, m117002y	YMAR-11700.02~M, ~Y
m560012m, m560012y	YMAR-5600.01.02~M, ~Y		

Codes for Created Variable

Marital/Cohabitation Status

1 = never married, cohabiting
2 = never married, not cohabiting
3 = married, spouse present
4 = married, spouse absent
5 = separated, cohabiting
6 = separated, not cohabiting
7 = divorced, cohabiting
8 = divorced, not cohabiting
9 = widowed, cohabiting
10 = widowed, not cohabiting

Collapsed Marital Status

0 = never married
1 = married
2 = separated
3 = divorced
4 = widowed

This program creates two variables that describe marital status/cohabitation status as of the interview date for respondents age 16 and older. Other respondents are valid skips (-4). Note that later partners take precedence over earlier partners.

The program also creates variables that provide the dates of the youth's first marriage and/or cohabitation in both a continuous month scheme and as actual dates (for more information on the continuous month scheme, see appendix 7 in this document). Summary variables count the total number of marriages and cohabitations for each youth. Note that these variables are available only for youths age 16 and older as of 12/31/99. If a respondent is cohabiting and then marries it is considered both a cohabitation and a marriage. If someone refuses or doesn't know the full date of their marriage or cohabitation, then the spell is counted in the total variables and the date variables are coded -1 or -2 as applicable.

```
/**Create flag indicating whether r. said there was a
spouse in the house in the hh roster**/
spo_hous=0;

array hhrel (i) hhrel01-HHREL14;
do i=1 to 14;
  if hhrel=1 or hhrel=2 then spo_hous=1;
end;

*Create a variable indicating r.'s marital status as they
report it;

if m650 ge 0 then marques=m650;
else marques=m620;

/*Set up initial marital status using dli created variables. */
prevmsta=MARSTAT3;     prevcmst=CMARSTA3;
mar_ttl=mar_ttl3;      coh_ttl=coh_ttl3;
dlimary= FMAR_Yr3;     dlimarm= FMAR_Mr3;
dlimarcm=fmarmr3;      dlicohm= FCOH_Mr3;
dlicohy= FCOH_Yr3;     dlicohcm=FCOHCMr3;

if cmarsta3=-5 then do; /*round 3 noninterview, go to
      round 2*/
  prevmsta=marstat2;       prevcmst=cmarsta2;
  mar_ttl=mar_ttl2;        coh_ttl=coh_ttl2;
  dlimary= FMAR_Yr2;       dlimarm= FMAR_Mr2;
  dlimarcm=fmarmr2;        dlicohm= FCOH_Mr2;
  dlicohy= FCOH_Yr2;       dlicohcm=FCOHCMr2;
end;

if cmarsta2=-5 and cmarsta3=-5 then do; /*round 3 and
      round 2 noninterview, go to round 1*/
  prevmsta=marstat1;       prevcmst=cmarsta1;
  mar_ttl=mar_ttl1;        coh_ttl=coh_ttl1;
  dlimary= FMAR_Yr1;       dlimarm= FMAR_Mr1;
  dlimarcm=fmarmr1;        dlicohm= FCOH_Mr1;
  dlicohy= FCOH_Yr1;       dlicohcm=FCOHCMr1;
end;

*Create interview date and previous interview date in
      continuous months and months/years;
if int_y gt 0 and int_m gt 0 then do;
   doicm=(int_y-1980)*12+int_m;
  end;
else if int_y eq -5 or int_m eq -5 then do;
   doicm=-5;
  end;
else if int_y in(-1,-2, -3,-4) or int_m in(-1, -2, -3, -4)
      then do;
   doicm=-3;
end;

if int_yr3 gt 0 and int_mr3 gt 0 then do;
   dlicm=(int_yr3-1980)*12+int_mr3;
end;
else if int_yr3 = -5 and int_yr2>-5 then do;
   dlicm=(int_yr2-1980)*12+int_mr2;
end;
else if int_yr2 = -5 and int_yr3 = -5 then do;
   dlicm=(int_yr1-1980)*12+int_mr1;
end;
else if int_yr3 in(-1,-2, -3,-4) or int_mr3 in(-1, -2, -3, -4)
      then do;
   dlicm=-3;
end;
else if int_yr2 in(-1,-2, -3,-4) or int_mr2 in(-1, -2, -3, -4)
      then do;
   dlicm=-3;
end;

/*set up arrays with marital/cohab status for each of the
      continuous months*/
array m     (l)   m001-m260;          /* married y/n */
array coh   (l)   coha001-coha260;    /* cohab y/n */
array mars  (l)   mars001-mars260;    /* marital status */

/* Initialize values to 0 for all who go through section,
      between dli and current interview date. */
if m700>-4 then do; marstat=2; ttlm=0; ttlc=0; end;

do l=1 to 260;
  if dlicm le L le doicm then do; m=0; coh=0; mars=2;
      end;
end;

/* In the array section, an index by "p" indicates a loop
      over partners, while an index by "t" indicates a loop
      over time periods. */
/*year start cohabitating*/
  array ysca (t) m31001y m700011y m700012y
      m700013y;
  array yscb (t) m31002y m700021y m700022y
      m700023y;
  array yscc (t) m31003y m700031y m700032y
      m700033y;
```

```
  array yscd (t) m31004y m700041y m700042y
        m700043y;

/*month start cohabitating*/
  array msca (t) m31001m m700011m m700012m
        m700013m;
  array mscb (t) m31002m m700021m m700022m
        m700023m;
  array mscc (t) m31003m m700031m m700032m
        m700033m;
  array mscd (t) m31004m m700041m m700042m
        m700043m;

        /*collapse all start years and months into arrays*/
          array ysc   (p) ysca yscb yscc yscd;
          array msc   (p) msca mscb mscc mscd;

/*legally married when began cohabitating*/
  array mbega (t) m45001 m4500a2 m4500a3 m4500a4;
  array mbegb (t) m45002 m4500b2 m4500b3 m4500b4;
  array mbegc (t) m45003 m4500c2 m4500c3 m4500c4;
  array mbegd (t) m45004 m4500d2 m4500d3 m4500d4;

        /*collapse married into array*/
          array mbeg  (p) mbega mbegb mbegc mbegd;

/*dummy for cohabitation continuous - since dli*/
  array cdlia (t) m46701 cdlia2 cdlia3 cdlia4;
  array cdlib (t) cdlib1 cdlib2 cdlib3 cdlib4;
  array cdlic (t) cdlic1 cdlic2 cdlic3 cdlic4;
  array cdlid (t) cdlicd cdlid2 cdlid3 cdlid4;

/*dummy for cohabitation continuous - since start (after
        dli date)*/
  array cbega (t) m47001 m910011 m910012 m910013;
  array cbegb (t) m47002 cbegb2 cbegb3 cbegb4;
  array cbegc (t) m47003 cbegc2 cbegc3 cbegc4;
  array cbegd (t) m47004 cbegd2 cbegd3 cbegd4;

        /* continuous cohab vars into array*/
          array cdli  (p) cdlia cdlib cdlic cdlid;
          array cbeg  (p) cbega cbegb cbegc cbegd;

/*year end cohabitating (not continuous)*/
  array yeca  (t) m48001y m920011y m920012y
        m920013y;
  array yecb  (t) m48002y yecb2 yecb3 yecb4;
  array yecc  (t) m48003y yecc2 yecc3 yecc4;
  array yecd  (t) m48004y yecd2 yecd3 yecd4;

/*month end cohabitating (not continuous)*/
  array meca  (t) m48001m m920011m m920012m
        m920013m;
  array mecb  (t) m48002m mecb2 mecb3 mecb4;
  array mecc  (t) m48003m mecc2 mecc3 mecc4;
  array mecd  (t) m48004m mecd2 mecd3 mecd4;

        /*collapse all end years and months into arrays*/
          array yec   (p) yeca yecb yecc yecd;
          array mec   (p) meca mecb mecc mecd;

/* First or next change in status. Read "howa2" as change
        in marital status for partner a, period 2. */
  array howa1 (t) m540011 m540021 m9800111
        howa41;
  array howa2 (t) m540012 m540022 howa32 howa42;
  array howa3 (t) m7900111 howa32 howa33 howa43;
  array howb3 (t) m7900211 howb32 howb33 howb43;

        /*collapse into how status changed*/
          array how1  (p) howa1 howb1 howc1 howd1;
          array how2  (p) howa2 howb2 howc2 howd2;
          array how3  (p) howa3 howb3 howc3 howd3;
          array how4  (p) howa4 howb4 howc4 howd4;

/*month and year change is marry*/
  array hyma1 (t) m570011y m101111y hyma13 hyma14;
  array hmma1 (t) m570011m m101111m hmma23
        hmma24;
  array hyma2 (t) m570012y hyma22 hyma23 hyma24;
  array hmma2 (t) m570012m hmma22 hymm23
        hymm24;
  array hmma3 (t) m820111m hmma32 hmma33
        hmma34;
  array hyma3 (t) m820111y hyma32 hyma33 hyma34;
  array hmmb3 (t) m820211m hmmb32 hmmb33
        hmmb34;
  array hymb3 (t) m820211y hymb32 hymb33 hymb34;

        /*collaspse into month and year change is marry*/
          array hym1  (p) hyma1 hymb1 hymc1 hymd1;
          array hmm1  (p) hmma1 hmmb1 hmmc1 hmmd1;
          array hym2  (p) hyma2 hymb2 hymc2 hymd2;
          array hmm2  (p) hmma2 hmmb2 hmmc2 hmmd2;
          array hym3  (p) hyma3 hymb3 hymc3 hymd3;
          array hmm3  (p) hmma3 hmmb3 hmmc3 hmmd3;
          array hym4  (p) hyma4 hymb4 hymc4 hymd4;
          array hmm4  (p) hmma4 hmmb4 hmmc4 hmmd4;

/*month & year change=separation, divorce, annulment*/
  array hysa1 (t) m560011y hysa12 hysa13 hysa14;
  array hmsa1 (t) m560011m hmsa12 hmsa13 hmsa14;
  array hysa2 (t) m560012y hysa22 hysa23 hysa24;
  array hmsa2 (t) m560012m hmsa22 hmsa23 hmsa24;
  array hysa3 (t) m810111y hysa32 hysa33 hysa34;
  array hmsa3 (t) m810111m hmsa32 hmsa33 hmsa34;
  array hysa4 (t) m810112y hysa42 hysa43 hysa44;
  array hmsa4 (t) m810112m hmsa42 hmsa43 hmsa44;

        /*collapse into month and year change=separation,
        divorce, annulment*/
          array hys1  (p) hysa1 hysb1 hysc1 hysd1;
          array hms1  (p) hmsa1 hmsb1 hmsc1 hmsd1;
          array hys2  (p) hysa2 hysb2 hysc2 hysd2;
```

```
        array hms2  (p) hmsa2 hmsb2 hmsc2 hmsd2;
        array hys3  (p) hysa3 hysb3 hysc3 hysd3;
        array hms3  (p) hmsa3 hmsb3 hmsc3 hmsd3;
        array hys4  (p) hysa4 hysb4 hysc4 hysd4;
        array hms4  (p) hmsa4 hmsb4 hmsc4 hmsd4;

/* marital status at end of cohab period */
  array estata  (t) m63001 m1080011m m1080012m
       m1080013m;
  array estatb  (t) m63002 estatb2 estatb3 estatb4;
  array estatc  (t) m63003 estatc2 estatc3 estatc4;
  array estatd  (t) m63004 estatd2 estatd3 estatd4;

      /* collapse marital status into one array */
        array estat (p) estata estatb estatc estatd;

/* month and year relationship ended */
  array eyra  (t) m117001y eyra2 eyra3 eyra4;
  array eyrb  (t) m117002y eyrb2 eyrb3 eyrb4;
  array eyrc  (t) eyrc1    eyrc2 eyrc3 eyrc4;
  array eyrd  (t) eyrd1    eyrd2 eyrd3 eyrd4;
  array emoa  (t) m117001m emoa2 emoa3 emoa4;
  array emob  (t) m117002m emob2 emob3 emob4;
  array emoc  (t) emoc1    emoc2 emoc3 emoc4;
  array emod  (t) emod1    emod2 emod3 emod4;

      /*collapse all start years and months into arrays*/
        array eyr   (p) eyra eyrb eyrc eyrd;
        array emo   (p) emoa emob emoc emod;

/* current staus of relationship */
  array fstata (t) m114001 fstata2 fstata3 fstata4;
  array fstatb (t) m114002 fstatb2 fstatb3 fstatb4;
  array fstatc (t) m114003 fstatc2 fstatc3 fstatc4;
  array fstatd (t) m114004 fstatd2 fstatd3 fstatd4;

      /*collapse married into array*/
        array fstat (p) fstata fstatb fstatc fstatd;

/* Widow flag */
  array widowa (t) widowa1 widowa2 widowa3 widowa4;
  array widowb (t) widowb1 widowb2 widowb3 widowb4;
  array widowc (t) widowc1 widowc2 widowc3 widowc4;
  array widowd (t) widowd1 widowd2 widowd3
       widowd4;
      array widow  (p) widowa widowb widowc widowd;

/*continuous start month of cohabitation*/
  array csmca (t) csmca1 csmca2 csmca3 csmca4;
  array csmcb (t) csmcb1 csmcb2 csmcb3 csmcb4;
  array csmcc (t) csmcc1 csmcc2 csmcc3 csmcc4;
  array csmcd (t) csmcd1 csmcd2 csmcd3 csmcd4;
      array csmc  (p) csmca csmcb csmcc csmcd;

/*continuous end month of cohabitation*/
  array cemca (t) cemca1 cemca2 cemca3 cemca4;
  array cemcb (t) cemcb1 cemcb2 cemcb3 cemcb4;
  array cemcc (t) cemcc1 cemcc2 cemcc3 cemcc4;
  array cemcd (t) cemcd1 cemcd2 cemcd3 cemcd4;
      array cemc  (p) cemca cemcb cemcc cemcd;

/*continuous start month of marriage*/
  array csmma (t) csmma1 csmma2 csmma3 csmma4;
  array csmmb (t) csmmb1 csmmb2 csmmb3 csmmb4;
  array csmmc (t) csmmc1 csmmc2 csmmc3 csmmc4;
  array csmmd (t) csmmd1 csmmd2 csmmd3 csmmd4;
      array csmm  (p) csmma csmmb csmmc csmmd;

/*continuous start month of separation, etc*/
  array csmsa (t) csmsa1 csmsa2 csmsa3 csmsa4;
  array csmsb (t) csmsb1 csmsb2 csmsb3 csmsb4;
  array csmsc (t) csmsc1 csmsc2 csmsc3 csmsc4;
  array csmsd (t) csmsd1 csmsd2 csmsd3 csmsd4;
      array csms  (p) csmsa csmsb csmsc csmcd;

/* married at date of last interview */
  array mdlia (t) mdlia1 mdlia2 mdlia3 mdlia4;
  array mdlib (t) mdlib1 mdlib2 mdlib3 mdlib4;
  array mdlic (t) mdlic1 mdlic2 mdlic3 mdlic4;
  array mdlid (t) mdlid1 mdlid2 mdlid3 mdlid4;
      array mdli  (p) mdlia mdlib mdlic mdlid;

/*Collapsed marital status*/
  array cmarsa (t) cmarsa1 cmarsa2 cmarsa3 cmarsa4;
  array cmarsb (t) cmarsb1 cmarsb2 cmarsb3 cmarsb4;
  array cmarsc (t) cmarsc1 cmarsc2 cmarsc3 cmarsc4;
  array cmarsd (t) cmarsd1 cmarsd2 cmarsd3 cmarsd4;
  array cmars (p) cmarsa cmarsb cmarsc cmarsd;

/*number of cohabitations*/
  array nuca (t) nuca1 nuca2 nuca3 nuca4;
  array nucb (t) nucb1 nucb2 nucb3 nucb4;
  array nucc (t) nucc1 nucc2 nucc3 nucc4;
  array nucd (t) nucd1 nucd2 nucd3 nucd4;
      array nuc (p) nuca nucb nucc nucd;

/*number of marriages*/
  array numa (t) numa1 numa2 numa3 numa4;
  array numb (t) numb1 numb2 numb3 numb4;
  array numc (t) numc1 numc2 numc3 numc4;
  array numd (t) numd1 numd2 numd3 numd4;
      array num (p) numa numb numc numd;

  /*dummy set to one if start month cohabitate is invalid*/
  array fixmc1 (t) fixmc11 fixmc12 fixmc13 fixmc14;
  array fixmc2 (t) fixmc21 fixmc22 fixmc23 fixmc24;
  array fixmc3 (t) fixmc31 fixmc32 fixmc33 fixmc34;
  array fixmc4 (t) fixmc41 fixmc42 fixmc43 fixmc44;
      array fixmc  (p) fixmc1 fixmc2 fixmc3 fixmc4;

  /*dummy set to one if start month marry is invalid*/
  array fixmm1 (t) fixmm11 fixmm12 fixmm13 fixmm14;
  array fixmm2 (t) fixmm21 fixmm22 fixmm23 fixmm24;
  array fixmm3 (t) fixmm31 fixmm32 fixmm33 fixmm34;
```

```
  array fixmm4 (t) fixmm41 fixmm42 fixmm43 fixmm44;
      array fixmm  (p) fixmm1 fixmm2 fixmm3 fixmm4;

  /*dummy set to one if start year cohabitate is invalid*/
  array fixyc1 (t) fixyc11 fixyc12 fixyc13 fixyc14;
  array fixyc2 (t) fixyc21 fixyc22 fixyc23 fixyc24;
  array fixyc3 (t) fixyc31 fixyc32 fixyc33 fixyc34;
  array fixyc4 (t) fixyc41 fixyc42 fixyc43 fixyc44;
      array fixyc  (p) fixyc1 fixyc2 fixyc3 fixyc4;

  /*dummy set to one if start year marry is invalid*/
  array fixym1 (t) fixym11 fixym12 fixym13 fixym14;
  array fixym2 (t) fixym21 fixym22 fixym23 fixym24;
  array fixym3 (t) fixym31 fixym32 fixym33 fixym34;
  array fixym4 (t) fixym41 fixym42 fixym43 fixym44;
  array fixym  (p) fixym1 fixym2 fixym3 fixym4;

/* This flag prevents the dli from being imputed as the c.v
      for marriage month. Since the program assigns the
      dli to the start date of marriage, this is not what
      should go into the c.v. */
  array flagcm1 (t) flagcm11 flagcm12 flagcm13 flagcm14;
  array flagcm2 (t) flagcm21 flagcm22 flagcm23 flagcm24;
  array flagcm3 (t) flagcm31 flagcm32 flagcm33 flagcm34;
  array flagcm4 (t) flagcm41 flagcm42 flagcm43 flagcm44;
      array flagcm  (p) flagcm1 flagcm2 flagcm3 flagcm4;

/* Is this continuous end month of spearation? Is it not
      found anywhere else in the program. */
  array cems  (p) cemsa cemsb cemsc cemsd;

/* Begin by coding respondents who have been
      cohabitating or married since dli. Set begin date of
      cohab. (or marriage) as dli. */

/* For respondents who report no beginning date of
      cohab. because they were cohabitating at dli, the
      start date of cohab must be created for the program.
      This is done outside the loop because the dli partner
      must be partner "a" and begin at loop 1. For the
      program, the start date of cohab is set to dli. The
      same is done for marriages. */
if m714=1 and m30501=1 then csmca1=dlicm;
if prevcmst=1 and m714=1 and m30501=1 then do;
   csmma1=dlicm;
   mdlia1=1;
   cmarsa1=1;
   flagcma1=1;
end;

/* These respondents have gotten married and during
      Round 4 but do not have a start month of
      separation because of the strange way they go
      through the loop section. */
if id in (1309501, 1550021, 1782191, 1787411,
      1928761, 1932841) then do;
   csmsa1=doicm; handedit=1;
end;

do p=1 to 4;
  do t=1 to 4;

    if cdli=1 then do;
      csmc=dlicm;
      flagcm=1;
    end;

/* Code respondents divorced at dli, using dli c.v. Set
      begin date of separation as dli. */
    if mdli=3 then do;
      cmars=3;
      csms=dlicm;
      flagcm=1;
    end;

/* If married at dli is don't know or refuse, then set
      collapsed marital status to -1 or -2. */
    if mdli in (-1,-2) then cmars=mdli;

/* Create continuous start month cohab variable from
      data, create dummies for negative answers. Any
      negative answer is set to the dli and dummies (fixyc,
      fixmc) are created to identify these cases. */
    if msc in (-1,-2,-3) then do;
      fixmc=1;
      csmomiss=1;
    end;

    /* Hand edit bad start month to January, if valid start
      year is given. */
    if msc in (-1,-2,-3) and ysc>0 then do;
      msc=1;
    end;

    if ysc>0 and msc>0 then csmc=(ysc-1980)*12 + msc;

/* If a start month is missing, set the start month to Jan.
      (coded above) If a start year is missing, set to dli. */
    if ysc in (-1,-2,-3) then do;
      csmc=dlicm; fixyc=1; csyrmiss=1;
    end;

/* Create continuous end month cohab variable from
      data. If continuous end month of cohabitation is
      missing, not cohabiting at dli (cdli ne 1) and
      cohabiting continuously with new partner (cbeg ne
      0), then ymec=int date. */
    /* Hand edit bad end month to Dec, if valid start year
      is given. */
    if mec in (-1,-2,-3) and yec>0 then do; mec=12; end;
    if yec>0 and mec>0 then cemc=(yec-1980)*12 + mec;
    if yec in (-1,-2,-3) then do; cemc=doicm; ceyrmiss=1;
      end;
    if mec in (-1,-2,-3) then do; cemomiss=1; end;
```

```
   /* Set upper bound on imputed month values. */
   if mec in (-1,-2,-3) and cemc>doicm then do;
       cemc=doicm; end;

/* If cohab since dli or married at beginning of cohab,
       then set cohab end date as int date. */
   if cbeg=1 or cdli=1 then cemc=doicm;

/* If married when began cohab, then assume married and
       change date arrays and update marriage date flags. */
   if mbeg=1 or mdli=1 then do;
      cmars=1;
      csmm=csmc;
      if fixmc=1 then fixmm=1;
      if csmomiss=1 then msmomiss=1;
      if fixyc=1 then fixym=1;
      if csyrmiss=1 then msyrmiss=1;

      /* Set flagcm back to dot, since we now have a date
       for marriage, instead of dli. */
      flagcm=.;

      /*set separation date as the end month of cohab*/
      if cemc>0 then csms=cemc;

      /* Marriage ends in legal separation */
      if how1=3 then cmars=2;   if how2=3 then cmars=2;
      if how3=3 then cmars=2;   if how4=3 then cmars=2;

      /* Marriage ends in divorce */
      if how1=4 then cmars=3;   if how2=4 then cmars=3;
      if how3=4 then cmars=3;   if how4=4 then cmars=3;

      /* Marriage ends in divorce */
      if how1=5 then cmars=0;   if how2=5 then cmars=0;
      if how3=5 then cmars=0;   if how4=5 then cmars=0;
   end;

/* For respondentds whose marital status changes since
       dli, update marriage/separation/cohab dates and
       collapsed marital status. */
/* Change is marriage:  beginning of loop. */
   if how1=1 and hym1>0 and hmm1>0 then do;
      cmars=1; csmm=(hym1-1980)*12 + hmm1;
   end;
   if how1=1 and (-4<hym1<0 or -4<hmm1<0) then do;
      cmars=1; csmm=doicm;
   end;
/* Change is legal separation. */
   if how1=3 and hys1>0 and hms1>0 then do;
      cmars=2; csms=(hys1-1980)*12 + hms1;
   end;
   if how1=3 and (-4<hym1<0 or -4<hmm1<0) then do;
      cmars=2; csms=doicm; meyrmiss=1;
   end;
/* Change is divorce. */
   if how1=4 and hys1>0 and hms1>0 then do;
      cmars=3; csms=(hys1-1980)*12 + hms1;
   end;
   if how1=4 and (-4<hym1<0 or -4<hmm1<0) then do;
      cmars=3; csms=doicm; meyrmiss=1;
   end;
/* Change is annulment. */
   if how1=5 and hys1>0 and hms1>0 then do;
      cmars=0; csms=(hys1-1980)*12 + hms1;
   end;
   if how1=5 and (-4<hym1<0 or -4<hmm1<0) then do;
      cmars=0; csms=doicm; meyrmiss=1;
   end;

   if how1 in (-1,-2) then cmars=how1;

/*At this point the program repeats the above code (from
       "beginning of loop" for marriages 2-4 (how2-how4).
       Contact NLS User Services for more information.*/

/* This loop collects information from the marital status
       at the end of relationship period. Code: 0 = R and
       partner not married, 1 = R and partner married, 2 =
       R and partner separated, 3 = R and partner
       divorced, 4 = R widowed. */
   if estat>-4 then cmars=estat;

   /* YMAR-114000 (fstat in the arrays) collects current
       status of the relationship. */
/* fstat=annulment, which translates into never married in
       collapsed mar. status */
   if fstat=2 and eyr>0 and emo>0 then do;
     cmars=0; csms=(eyr-1980)*12 + emo;
   end;
   if fstat=2 and (-4<eyr<0 or -4<emo<0) then do;
     cmars=0; csms=doicm;
   end;
/* separation */
   if fstat=3 and eyr>0 and emo>0 then do;
     cmars=2; csms=(eyr-1980)*12 + emo;
   end;
   if fstat=3 and (-4<eyr<0 or -4<emo<0) then do;
     cmars=2; csms=doicm;
   end;
/* divorce */
   if fstat=4 and eyr>0 and emo>0 then do;
     cmars=3; csms=(eyr-1980)*12 + emo;
   end;
   if fstat=4 and (-4<eyr<0 or -4<emo<0) then do;
     cmars=3; csms=doicm;
   end;
/* widowed */
   if fstat=5 and eyr>0 and emo>0 and cmars=1 then do;
     cmars=4; csms=(eyr-1980)*12 + emo; cemc=(eyr-
      1980)*12 + emo; widow=1;
   end;
```

```
   if fstat=5 and (-4<eyr<0 or -4<emo<0) and cmars=1
      then do;
     cmars=4;                    csms=doicm;
     cemc=doicm;                 meyrmiss=1;
     ceyrmiss=1;                 widow=1;
   end;
   if -4<fstat<0 then cmars=fstat;

/* If there is no change in marital status since dli or
      beginning of marriage and the respondent reports
      being married at dli (cmars=1), then let the time of
      separation be the interview date. */
  if cmars=1 and (cdli=1 or cbeg=1) then csms=doicm;
   if csms>0 then cems=doicm;
   if csmm>0 then cemm=doicm;

/* Count the total number of cohabs. */
   if csmc>0 then do;
     ttlc=ttlc+1; nuc=ttlc;
     if nuc=1 then r4cohcm=csmc;
   end;
/* Count the total number of marriages */
   if csmm>0 then do;
     ttlm=ttlm+1; num=ttlm;
     if num=1 then r4marcm=csmm;
   end;
   if flagcm=1 then dlist=1;
   corrc=0;
   corrm=0;

/* Create a correction variable to prevent double counting
      the same marriage/cohab partner since we set the
      start months of marriage/cohab at dli if the person
      ha a partner at dli. */
   if ttlc>-1 or ttlm>-1 then do;
     if csmca2>0 and csmca2 ne . then corrc=corrc+1;
     if csmcb2>0 and csmcb2 ne . then corrc=corrc+1;
     if csmcc2>0 and csmcc2 ne . then corrc=corrc+1;
     if csmcd2>0 and csmcd2 ne . then corrc=corrc+1;
     if csmma2>0 and csmma2 ne . then corrm=corrm+1;
     if csmmb2>0 and csmmb2 ne . then corrm=corrm+1;
     if csmmc2>0 and csmmc2 ne . then corrm=corrm+1;
     if csmmd2>0 and csmmd2 ne . then corrm=corrm+1;
     if csmca3>0 and csmca3 ne . then corrc=corrc+1;
     if csmcb3>0 and csmcb3 ne . then corrc=corrc+1;
     if csmcc3>0 and csmcc3 ne . then corrc=corrc+1;
     if csmcd3>0 and csmcd3 ne . then corrc=corrc+1;
     if csmma3>0 and csmma3 ne . then corrm=corrm+1;
     if csmmb3>0 and csmmb3 ne . then corrm=corrm+1;
     if csmmc3>0 and csmmc3 ne . then corrm=corrm+1;
     if csmmd3>0 and csmmd3 ne . then corrm=corrm+1;
     if csmca4>0 and csmca4 ne . then corrc=corrc+1;
     if csmcb4>0 and csmcb4 ne . then corrc=corrc+1;
     if csmcc4>0 and csmcc4 ne . then corrc=corrc+1;
     if csmcd4>0 and csmcd4 ne . then corrc=corrc+1;
     if csmma4>0 and csmma4 ne . then corrm=corrm+1;
     if csmmb4>0 and csmmb4 ne . then corrm=corrm+1;
     if csmmc4>0 and csmmc4 ne . then corrm=corrm+1;
     if csmmd4>0 and smmd4 ne . then corrm=corrm+1;
   end;

/* Subtract the correction from the count variable to
      create the total number of marriages & total number
      of cohabs c.v. */
   ttlcnew=ttlc-corrc;
   ttlmnew=ttlm-corrm;

/*Create check variable for missing start/stop months of
      marriage and cohabitation. */
   if csmc>. and cemc=. then cerror=1;
   if csmc=. and cemc>. then cerror=1;
   if csmm>. and csms=. then merror=1;
   if csmm=. and csms>. then merror=1;

/*correct for rs who don't know start month of
      cohabiting*/
if fixmc=1 and (msc<0 and ysc>-4) then do;
   r4cohy=ysc; r4cohcm=-3; r4cohm=-3;
end;

/*correct for rs who don't know start year of cohabiting*/
if fixyc=1 and (ysc<0 and msc>-4) then do;
   r4cohm=msc; r4cohcm=-3; r4cohy=-3;
end;

/* correct for rs who don't know start year and month of
      cohabiting*/
if fixyc=1 and fixmc=1 then do;
   r4cohm=-3; r4cohcm=-3;r4cohy=-3;
end;

/*correct for rs who don't know start month of marriage*/
if fixmm=1 and (ysc<0 and msc>-4) then do;
   r4mary=ysc; r4marcm=-3; r4marm=-3;
end;

/*correct for rs who don't know start year of marriage*/
if  fixym=1 and (ysc<0 and msc>-4) then do;
   r4marm=msc;
   r4marcm=-3;
   r4mary=-3;
end;

/*correct for rs who don't know start year and month of
      marriage*/
if fixym=1 and fixmm=1 then do;
   r4marm=-3; r4marcm=-3; r4mary=-3;
end;

/* Now that the start & end dates of marriages/cohabs are
      created, we place that info into the array. The info
      from the last week is used to create the variable for
      marital status and collapsed marital status. */
   C=0;
```

```
  do L=1 to 260;
    C=C+1;
    if csmc>0 and cemc>0 and csmc <= C <= cemc then
      coh=1;
    if cmars=1 and csmm>0 and csms>0 and csmm le c le
      csms then m=1;
    if cmars in (2,3) and csms>0 and cems>0 and csms le
      c le cems then m=cmars;
    if cmars in (-2,-3) then m=cmars;
    if c=doicm then do;
      if m in (-1,-2) then mars=m;
      if m=0 and coh=1 then mars=1;
      if m=0 and coh=0 then mars=2;
      if m=1 and coh=1 then mars=3;
      if m=1 and coh=0 then mars=4;
      if m=2 and coh=1 then mars=5;
      if m=2 and coh=0 then mars=6;
      if m=3 and coh=1 then mars=7;
      if m=3 and coh=0 then mars=8;
      if m=4 and coh=1 then mars=9;
      if m=4 and coh=0 then mars=10;
      marstat=mars;
    end;
  end;
 end;
end;

/*Create collapsed marital status from marital status. */
if 1 le marstat le 2 then cmarstat=0;
if 3 le marstat le 4 then cmarstat=1;
if 5 le marstat le 6 then cmarstat=2;
if 7 le marstat le 8 then cmarstat=3;
if 9 le marstat le 10 then cmarstat=4;
if -2 le marstat le -1 then cmarstat=marstat;

/* For 10 respondents, their Round 3 c.v. for first
      marriage date post-dates a marriage they report in
      Round 4. We accept the Round 3 data as true, since
      it is closer to the time of marriage/cohabitation. */
if dlimarcm>0 then marcm=dlimarcm;
if r4marcm>0 then marcm=r4marcm;
if dlimarcm>0 and r4marcm>0 then marcm=dlimarcm;
if dlicohcm>0 then cohcm=dlicohcm;
if r4cohcm>0 then cohcm=r4cohcm;
if dlicohcm>0 and r4cohcm>0 then cohcm=dlicohcm;
if -4<r4marcm<0 and dlimarcm=-4 then
      marcm=r4marcm;
if r4marcm=-4 and 0<dlimarcm<-4 then
      marcm=dlimarcm;
if -4<r4cohcm<0 and dlicohcm=-4 then cohcm=r4cohcm;
if r4cohcm=-4 and 0<dlicohcm<-4 then
      cohcm=dlicohcm;

/* Using continuous month information, create month
      and year of marriage and cohabitation. */
array cvcm cohcm marcm;
array cvy cohy mary;
array cvm cohm marm;

do over cvcm;
  if 253 le cvcm le 264 then do; cvy=2001; cvm=cvcm-
      252; end;
  if 241 le cvcm le 252 then do; cvy=2000; cvm=cvcm-
      240; end;
  if 229 le cvcm le 240 then do; cvy=1999; cvm=cvcm-
      228; end;
  if 217 le cvcm le 228 then do; cvy=1998; cvm=cvcm-
      216; end;
  if 205 le cvcm le 216 then do; cvy=1997; cvm=cvcm-
      204; end;
  if 193 le cvcm le 204 then do; cvy=1996; cvm=cvcm-
      192; end;
  if 181 le cvcm le 192 then do; cvy=1995; cvm=cvcm-
      180; end;
  if 169 le cvcm le 180 then do; cvy=1994; cvm=cvcm-
      168; end;
  if 157 le cvcm le 168 then do; cvy=1993; cvm=cvcm-
      156; end;
  if cvcm=. then do; cvcm=-4; cvy=-4; cvm=-4; end;
  if cvcm=-3 then do; cvcm=-3; cvy=-3; cvm=-3; end;
end;

/*if old date present and don't deny then use old date*/
if dlicohcm ne -4 and (m712=1 or m714=1 or m716=0 or
      (m712=0 and m718=1)) then do; cohcm=dlicohcm;
      cohy=dlicohy; cohm=dlicohcm; end;
if dlimarm gt -4 and (m712=1 or m714=1 or m716=0)
      then do; marcm=dlimarcm; mary=dlimary;
      marm=dlimarm; end;

/*previously in relationship, in more since interview*/
/* These respondents had a relationship prior to dli, then
      a new relationship during the R4 interview period.
      Here we simply add the previous c.v. to the Round
      4 count of cohabs/marriages for the toal c.v. */
if m712=1 and m740=1 and ttlmnew=>0 and mar_ttl=>0
      then do; ttlmnew=ttlmnew+mar_ttl; end;
if m712=1 and m740=1 and ttlcnew=>0 and coh_ttl=>0
      then do; ttlcnew=ttlcnew+coh_ttl; end;

/*in relationship at int date, no more since interview*/
/* These respondents had a relationship at dli and no
      other relationships since. This includes people who
      lived and/or married to the same person throughout
      and people whose relationship ended in the R4
      interview period. By definition, their total cohab.
      and total marriage c.v. should be the same as their
      dli c.v.'s. */
if m714=1 and m730=0 and coh_ttl>-4 then do;
      ttlcnew=coh_ttl; end;
if m714=1 and m730=0 and mar_ttl>-4 then do;
      ttlmnew=mar_ttl; end;

/*in relationship at int date, in more since interview*/
```

```
/* These respondents have relationships at dli, and have
      other relationships during the interview period. The
      counter variable setup corrc will erase one of these
      relationships, but this is the one case where it
      shouldn't. There are 6 of these people. */
if m714=1 and m730=1 and ttlmnew=>0 and mar_ttl=>0
      then do; ttlmnew=ttlmnew+mar_ttl; end;
if m714=1 and m730=1 and ttlcnew=>0 and coh_ttl=>0
      then do; ttlcnew=ttlcnew+coh_ttl; end;

***Change marstat if r. said s/he was married and said
      there was a spouse in hh, but misunderstood later
      questions about changes in marital status;
/* These repsondents indicate they are married (YMAR-
      620) and have a spouse present in the HH (see HH
      roster), but give no date for marriage. There is a
      chance the skip pattern confuses the respondent,
      thinking they already told us about the marriage at
      YMAR-620 and then saying there are no changes in
      the marital status, which is where the marriage date
      would come from. The first marriage date c.v.'s are
      set to -3, the marital stauts (collapased and regular)
      is changed to show marriage, the total number of
      marriages is set to -3, and the cohab dates and
      counter are unchanged. The total number of
      marriages is carried over from the previous round. */
   ***For pubid=3955, log said partner was away in
      treatment so she really was married;
   **Exclude pubid=8545 because log says not legally
      married;
   fixflag=0;
   if cmarstat eq 0 and m620=1 and spo_hous = 1 /* and
      pubid ne 8545 */ then do;
        cmarstat=1;
        if marstat=1 then marstat=3;
        if marstat=2 then marstat=4;
        fixflag=1;                ttlmnew=mar_ttl;
        ttlcnew=coh_ttl;          marcm=dlimarcm;
        mary=dlimary;             marm=dlimarm;
   end;

/* The respondents who refuse questions throughout the
      marriage section are coded as -3 for marital status,
      collapsed marital status, and total number of
      marriages/cohabs. First date of marriage/cohab is
      whatever was in the previous round. */
if m45001=-1 or m45002=-1 or M7900111 in (-1,-2)
      then do; marstat=-3; cmarstat=-3; ttlmnew=-3; end;
if m740 in (-1,-2) then do; marstat=-3; cmarstat=-3;
      ttlmnew=-3; ttlcnew=-3; end;

/* The remaining merror people (9 of them - there are no
      cerror) have a start date of separation but no start
      date of marriage. since it is difficult to tell whether
      these people were married to begin with, we are
      coding them as -3. */
if merror=1 or cerror=1 then do; marstat=-3; cmarstat=-3;
      ttlmnew=-3; end;
if m700=-4 then do; marstat=-4; cmarstat=-4;
      ttlmnew=-4; ttlcnew=-4; cohcm=-4;
      marcm=-4;cohy=-4; mary=-4; cohm=-4; marm=-4;
      end;
if m700=-5 then do; marstat=-5; cmarstat=-5;
      ttlmnew=-5; ttlcnew=-5; cohcm=-5; marcm=-5;
      cohy=-5; mary=-5; cohm=-5; marm=-5; end;
if marstat in (-1,-2) then marstat=-3;
if cmarstat in (-1,-2) then cmarstat=-3;
if cohcm in (-1,-2) then cohcm=-3;
if marcm in (-1,-2) then marcm=-3;
if cohm in (-1,-2) then cohm=-3;
if cohy in (-1,-2) then cohy=-3;
if marm in (-1,-2) then marm=-3;
if mary in (-1,-2) then mary=-3;

endsas;
```

YOUTH'S FERTILITY AND CHILD STATUS

Variables Created:

CV_CHILD_BIRTH_DATE.xx_M
CV_CHILD_BIRTH_DATE.xx_Y
CV_CHILD_DEATH_DATE.xx_M
CV_CHILD_DEATH_DATE.xx_Y
CV_CHILD_BIRTH_MONTH.xx
CV_CHILD_DEATH_MONTH.xx
CV_CHILD_STATUS.xx
CV_BIO_CHILD_NR
CV_BIO_CHILD_HH

Variables Used

Name in Program	Question Name on CD	Name in Program	Question Name on CD
oldsta01- oldsta04	YFER-1850.01-YFER-1850.04	bm03, by03	BIOCHILD_BDATE.03~M, ~Y
dmold, dyold	YFER-1860.01~M, ~Y	bioid01- bioid03	BIOCHILD_ID.01- BIOCHILD_ID.03
bmnew01, bynew01	YFER-5600.01~M, ~Y	biouid01- biouid03	BIOCHILD_UID.01- BIOCHILD_UID.03
bmnew02, bynew02	YFER-5600.02~M, ~Y	id	PUBID
newsta01, newsta02	YFER-5900.01, .02	bmold01, byold01	TOTBIOCHILDUPD2_BDATE.01~M, ~Y
dmnew, dynew	YFER-6000.01~M, ~Y	bmold02, byold02	TOTBIOCHILDUPD2_BDATE.02~M, ~Y
bm01, by01	BIOCHILD_BDATE.01~M, ~Y	bmold03, byold03	TOTBIOCHILDUPD2_BDATE.03~M, ~Y
bm02, by02	BIOCHILD_BDATE.02~M, ~Y	pubid	PUBID

Codes for Created Variables

Date of birth and death variables

Date variables are presented as both the actual month and year and the month number in a continuous month scheme.

Status variables

1 Adopted
2 Deceased
3 Non-resident, foster care
4 Non-resident, not adopted or in foster care
5 Resident

This program creates a number of variables describing the youth's fertility and the current status of the youth's children. For more information on the continuous month system, see appendix 7 in this document.

```
/* first, create a variable indicating the bmonth(i), birth
      month, and byear(i), birth year, for each biological
      child. Use variables from roster: BIOCHILD, which
      is the "clean" roster for all bio children */
array bm(3) bm01 bm02 bm03;
array by(3) by01 by02 by03;
array bmonth(3)  bmonth01 bmonth02 bmonth03;
array byear(3)   byear01 byear02 byear03;

/*small problem with recoding*/
if id=1735931 and by01=-2 then do;
  bm01=-3; by01=-3;
end;
/*correct the bdate when report a differing date of birth
      and age (HHIFINAL) in r4 (based on R5 data)*/
if id=1353832 then do;
  bm01=12; by01=1999; bmold01=12; byold01=1999;
end;
if id=1139502 then do;
  bm02=-3; by02=-3; bmold02=-3; byold02=-3;
      oldsta02=1;
end;
if id=1639431 then do;
  bm02=-4; by02=-4;
end;
/*don't know the bdate when report an early date of birth
      (pre-1990) and a differing age (HHIFINAL) in r4*/
if id=1647561 then do;
  bm01=-3; by01=-3; bmold01=-3; byold01=-3;
end;
/*correct the YFER report conflicts with BIOCHILD
      report*/
if id=1223531 then do;
  bm02=9; by02=2000;
end;
if id=1722971 then do;
  bmold02=6; byold02=1998; oldsta02=1;
end;
/*problem date resolved in r4 comments*/
if id=1830422 then byold02=2000;

do i=1 to 3; bmonth(i)=bm(i); byear(i)=by(i);
end;

/* second, create a continuous month scheme variable for
      the month of birth of the children using the
      formula: (12*(byear(i)-1980)+bmonth(i)) */
array mob(3) mob01 mob02 mob03;
do i=1 to 3;
 mob(i)=-4;
 if bmonth(i) eq -3 or byear(i) eq -3 then mob(i)=-3;
```

```
  if bmonth(i) gt 0 and byear(i) gt 0 then
        mob(i)=12*(byear(i)-1980)+bmonth(i);
  if bmonth(i) eq -5 then mob(i)=-5;
end;

/* third, create an actual date variable for the date of
        death of the youth's children. For old reported
        babies, use TOTBIOCHILDUPD2 roster birthday
        and match up with birthday in BIOCHI roster,
        yfer1860 variables are used to get the date of death.
        For new reported babies, use new baby birthday
        from the fertility section(yfer5600) and match up
        with birthday in BIOCHI roster, yfer6000 variables
        are used to get the date of death. */
array bmnew(3) bmnew01 bmnew02 bmnew03;
array bynew(3) bynew01 bynew02 bynew03;
array bmold(3) bmold01 bmold02 bmold03;
array byold(3) byold01 byold02 byold03;
array bioid(3) bioid01 bioid02 bioid03;
array dmonth(3) dmonth01 dmonth02 dmonth03;
array dyear(3) dyear01 dyear02 dyear03;
array newsta(3) newsta01 newsta02 newsta03;
array oldsta(3) oldsta01 oldsta02 oldsta03;

do i=1 to 3;
  dmonth(i)=dmold; dyear(i)=dyold;
  if bmonth(i)=-5 then dmonth(i)=-5;
  if byear(i)=-5 then dyear(i)=-5;
end;

do i=1 to 3;
  if (bm(i)=bmold01 and by(i)=byold01) then do;
     dmonth(i)=dmold; dyear(i)=dyold;
  end;
  if (bm(i)=bmnew01 and by(i)=bynew01) then do;
     dmonth(i)=dmnew; dyear(i)=dynew;
  end;
end;

/* fourth, create a continuous month scheme variable for
        the month of death of the children using the
        formula: (12*(dody(i)-1980)+dodm(i)) */
array mod(3) mod1 mod02 mod3;

do i=1 to 3;
  if dmonth(i)=-4 then mod(i)=-4;
  if dmonth(i)=-3 then mod(i)=-3;
  if dmonth(i) gt 0 and dyear(i) ge 1980 then
        mod(i)=12*(dyear(i)-1980)+dmonth(i);
  if dmonth(i)=-5 then mod(i)=-5;
end;

/* fifth, create a variable indicating the status of youth's
        first (second, third) child:
The old reported baby's status is collected through
        yfer1850, and the new reported baby's status is
        collected through yfer5900 */
array status(3) status01 status02 status03;

/*no match on bmold and byold for some r's due to
        coding problem & changed here*/
if id=1735931 or id=1032671 then do;
   bmold01=-3; byold01=-3;
end;

/* coding status according to the new questionaire */
do i=1 to 3;
  if (bm(i)=bmold01 and by(i)=byold01) then do;
     status(i)=oldsta01;
  end;
  if (bm(i)=bmold02 and by(i)=byold02) then do;
     status(i)=oldsta02;
  end;
  if (bm(i)=bmold03 and by(i)=byold03) then do;
     status(i)=oldsta03;
  end;
end;

do i=1 to 3;
  if (bm(i)=bmnew01 and by(i)=bynew01) then do;
     status(i)=newsta01;
  end;
  if (bm(i)=bmnew02 and by(i)=bynew02) then do;
     status(i)=newsta02;
  end;
end;

/* for kids who don't have a birthdate on the
        totbioupdate roster, this should not happen in
        round 5*/
do i=1 to 3;
   if status01=. and status02=-4 and status03=-4 and
        oldsta01 gt 0 then do;
      status01=oldsta01;
   end;
   if status01=. and status02=-4 and status03=-4 and
        oldsta01 gt -4 and oldsta01 lt 0 then do;
      status01=-3;
   end;
   if status01=. and status02=-4 and status03=-4 and
        newsta01 gt 0 then do;
      status01=newsta01;
   end;
   if status01=. and status02=-4 and status03=-4 and
        newsta01 gt -4 and newsta01 lt 0 then do;
      status01=-3;
   end;
   if status02=. and status03=-4 and oldsta02 gt 0 then do;
      status02=oldsta02;
   end;
   if status02=. and status03=-4 and oldsta02 gt -4 and
        oldsta02 lt 0 then do;
      status02=-3;
   end;
```

```
  if status02=. and status03=-4 and newsta02 gt 0 then
      do;
    status02=newsta02;
  end;
  if status02=. and status03=-4 and newsta02 gt -4 and
      newsta02 lt 0 then do;
    status02=-3;
  end;
end;

if id=1421191 then do;
  status01=1; status02=1;
end;

/* recoding status as we did in the first 3 rounds: 1 to 5 as
      stated above */
do i=1 to 3;
 if status(i)=1 then status(i)=5;
 else if status(i)=2 then status(i)=4;
 else if status(i)=3 then status(i)=1;
 else if status(i)=4 then status(i)=3;
 else if status(i)=5 then status(i)=4;
 else if status(i)=6 then status(i)=5;
 else if status(i)=7 then status(i)=5;
 else if status(i)=8 then status(i)=4;
 else if status(i)=9 then status(i)=2;
 else if status(i)=10 then status(i)=4; /*always double
      check these cases*/
 else if status(i)=11 then status(i)=3;
 else if status(i)=12 then status(i)=4;
 else if status(i)=999 then status(i)=-3;

/* picking up two missing cases from this questionaire */
 else if dmonth(i) gt 0 then status(i)=2;
 else if status(i)=-4 then status(i)=-4;
 else if status(i)=-5 then status(i)=-5;
end;

if id=1462711 then status01=4;
if id=1278121 then status01=5;

/* sixth, the number of children ever born and residing in
      the household (tbiores) */
array biores(3) biores1 biores2 biores3;

do i=1 to 3;
 biores(i)=0;
 if status(i) eq 5 then biores(i)=1;
end;

tbiores=biores1+biores2+biores3;
if (mob01=-4 and mob02=-4 and mob03=-4) then
      tbiores=-4;
if (status01=-3 or status02=-3 or status03=-3) then
      tbiores=-3;
if mob01=-5 then tbiores=-5;

/* seventh, the number of children ever born and not
      residing in the household (tbionres) */
array bionres(3) bionres1-bionres3;

do i=1 to 3;
 bionres(i)=0;
 if (status(i) eq 1 or status(i) eq 3 or status(i) eq 4) and
      status(i) ne 2 then bionres(i)=1;
end;

tbionres=bionres1+bionres2+bionres3;
if (mob01=-4 and mob02=-4 and mob03=-4) then
      tbionres=-4;
if (status01=-3 or status02=-3 or status03=-3) then
      tbionres=-3;
if mob01=-5 then tbionres=-5;

/* Last, sort created variables by birthdays, so that the first
      child listed is the oldest child. */
array smob(3) smob1 smob2 smob3;

do i=1 to 3; smob(i)=mob(i); end;
do i=1 to 3; if mob(i)=-3 then smob(i)=1000; end;

m1=-4; m2=-4; m3=-4;

/* consider the families with one child */
if (smob1>-4 and smob2=-4 and smob3=-4) or (smob1=-
      4 and smob2>-4 and smob3=-4) or (smob1=-4 and
      smob2=-4 and smob3>-4) then do;
 if smob1>-4 then m1=smob1;
 if smob2>-4 then m1=smob2;
 if smob3>-4 then m1=smob3;
end;

/* consider families with 2 children */
if (smob1>-4 and smob2>-4 and smob3=-4) then do;
 if smob1 ge smob2 then do;
   m1=smob2; m2=smob1;
 end;
 if smob1<smob2 then do;
   m1=smob1; m2=smob2;
 end;
end;

if (smob1=-4 and smob2>-4 and smob3>-4) then do;
 if smob2 ge smob3 then do; m1=smob3; m2=smob2;
      end;
 if smob2<smob3 then do; m1=smob2; m2=smob3; end;
end;

if (smob1>-4 and smob2=-4 and smob3>-4) then do;
 if smob1 ge smob3 then do; m1=smob3; m2=smob1;
      end;
 if smob1<smob3 then do; m1=smob1; m2=smob3; end;
end;
```

```
/* consider families with three children */
if (smob1>-4 and smob2>-4 and smob3>-4) then do;
  m1=min(smob1, smob2, smob3);
  m3=max(smob1, smob2, smob3);
end;

do i=1 to 3;
  if smob(i) ne m1 and smob(i) ne m3 then m2=smob(i);
end;

if smob1=-5 then m1=-5; if smob3=-5 then m3=-5;

array cv00bm(3) cv00bm1 cv00bm2 cv00bm3;
array cv00by(3) cv00by1 cv00by2 cv00by3;
array cv00mob(3) cv00mob1 cv00mob2 cv00mob3;
array cv00dm(3) cv00dm1 cv00dm2 cv00dm3;
array cv00dy(3) cv00dy1 cv00dy2 cv00dy3;
array cv00mod(3) cv00mod1 cv00mod2 cv00mod3;
array cv00stat(3) cv00sta1 cv00sta2 cv00sta3;

do i=1 to 3;
   if m1=smob(i) then do;
      cv00bm1 =bmonth(i);  cv00by1 =byear(i);
      cv00mob1=mob(i);     cv00dm1 =dmonth(i);
      cv00dy1 =dyear(i);   cv00mod1=mod(i);
      cv00sta1=status(i);
   end;
end;

do i=1 to 3;
   if m2=smob(i) then do;
      cv00bm2 =bmonth(i);  cv00by2 =byear(i);
      cv00mob2=mob(i);     cv00dm2 =dmonth(i);
      cv00dy2 =dyear(i);   cv00mod2=mod(i);
   cv00sta2=status(i);
   end;
end;

do i=1 to 3;
   if m3=smob(i) then do;
      cv00bm3 =bmonth(i);  cv00by3 =byear(i);
      cv00mob3=mob(i);     cv00dm3 =dmonth(i);
      cv00dy3 =dyear(i);   cv00mod3=mod(i);
      cv00sta3=status(i);
   end;
end;

endsas;
```

NUMBER OF RESIDENCES SINCE AGE 12

Variables Created: CV_TTL_RESIDENCES

Variables Used

Name in Program	Question Name on CD
cvr1	CV_TTL_RESIDENCES
cvr2	CV_TTL_RESIDENCES
cvr3	CV_TTL_RESIDENCES
y35004	YHHI-3500
y36004	YHHI-3600
pubid	PUBID

This program calculates the number of residences in which youth has lived since age 12. In round 1, the variable was created with information from the parent interview. In round 2, information collected from the respondent was combined with the round 1 variable to update the previous information. Subsequent rounds continue to update the variable with respondent-provided data.

```
resid=-4;

/*R interviewed in R3 and R4*/
if cvr3=-1 or y36004=-1 then resid=-1;/*Refusal in R3 or R4*/
if cvr3=-2 or y36004=-2 then resid=-2;/*DK in R3 or R4*/
if cvr3=-3 then resid=-3;/*Parents not interviewed*/
if cvr3 ge 0 and y36004 ge 0 then resid=CVR3+y36004;
if cvr3 ge 0 and y35004=0 then resid=cvr3;/*Instead (if cvr3 ge 0 and y36004=-4 then resid=cvr3;) would also have
yielded same results.*/

/*R interviewed in R4 but not in R3 and also taking care of the cases when not interviewed in R2 as well*/

if cvr3=-5 then do;
  if cvr2=-1 then resid=-1;
  if cvr2=-2 then resid=-2;
  if cvr2=-3 then resid=-3;
  if cvr2 ge 0 and y36004 ge 0 then resid=cvr2+y36004;
  if cvr2 ge 0 and Y35004=0 then resid=cvr2;
  if cvr2=-5 then do;
    if cvr1=-1 then resid=-1;
    if cvr1=-2 then resid=-2;
    if cvr1=-4 then resid=-3;
    if cvr1 ge 0 and y36004 ge 0 then resid=cvr1+y36004;
    if cvr1 ge 0 and Y35004=0 then resid=cvr1;
  end;
end;

/*R not interviewed in R4*/
if y35004=-5 then resid=-5;

endsas;
```

NLSY97 Appendix 4:

Geographic Variable Creation

Several variables in the main data set provide information about the respondent's area of residence. These variables permit researchers to identify key characteristics of the area without needing access to the geocode CD-ROM. Geographic variables were created using a software program called Maptitude; therefore, no programming code is provided for these variables. Instead, this document offers a brief general description of the methods used to generate these variables. For more information about the process of classifying a respondent's metropolitan area or about the geographic variables in general, refer to the introduction to the *Geocode Codebook Supplement* or contact NLS User Services.

> ***User Notes:*** Researchers should be aware that the process for geocoding respondent residences has been changed since the initial survey round. All residence information was re-geocoded using the new, more accurate approach, so all variables are comparable across rounds. However, researchers should not use geographic variables from old CDs in analyses; all geographic data should be taken from the newest CD.

CENSUS REGION OF RESIDENCE AT SURVEY DATE

Variable Created: CV_CENSUS_REGION

This variable classifies respondents as residing in one of four regions defined by the U.S. Census Bureau. These regions are as follows:

Census Division	States
Northeast	Connecticut, Maine, Massachusetts, New Hampshire, New Jersey, New York, Pennsylvania, Rhode Island, and Vermont
North Central	Illinois, Indiana, Iowa, Kansas, Michigan, Minnesota, Missouri, Nebraska, North Dakota, Ohio, South Dakota, and Wisconsin
South	Alabama, Arkansas, Delaware, District of Columbia, Florida, Georgia, Kentucky, Louisiana, Maryland, Mississippi, North Carolina, Oklahoma, South Carolina, Tennessee, Texas, Virginia, and West Virginia
West	Alaska, Arizona, California, Colorado, Hawaii, Idaho, Montana, Nevada, New Mexico, Oregon, Utah, Washington, and Wyoming

MSA STATUS AT SURVEY DATE

Variable Created: CV_MSA

This variable provides users with information about whether the respondent lived in the central city of the MSA, in another part of the MSA, or outside of an MSA. As defined by the Census Bureau, a central city is the major city lying within a Metropolitan Statistical Area (MSA). Initially, a variable was created using the TIGER/Line files (a database developed by the Census Bureau; the round 4 variable used the 1998 version) to determine whether the respondent lived in an MSA. A second variable was created based on "places" data in the Maptitude program that identified whether the respondent lived in the central city. The version of Maptitude used for round 4 used the 1992 TIGER/Line files to classify places as central cities. The variables were then combined to produce a single MSA/central city variable. For this dataset, respondents are coded as follows:

1 not in MSA
2 in MSA, not central city
3 in MSA, central city
4 in MSA, not known
5 not in country

RURAL VS. URBAN

Variable Created: CV_URBAN_RURAL

Places are identified as urban or rural by the Census Bureau. Urban places are those in "urbanized areas" or "places" with a population of at least 2,500; all other areas are rural. According to the Census Bureau, about 25 percent of the U.S. population lives in rural areas. The "urbanized area" map in the Maptitude software used in round 4 was based on the 1990 Census Bureau TIGER/Line files. Respondents residing in urbanized areas are coded 1 and those residing in rural areas are coded 0. Census Bureau information on urban and rural places can be retrieved from the following internet site: http://www.census.gov/geo/www/tiger/index.html.

Users should note that this variable includes an "unknown" category. This value is assigned to respondents whose zip code includes both urban and rural areas or whose residence cannot be identified precisely enough to classify it as urban or rural. Respondents without valid address data are assigned a value of –3, invalid skip. Respondents who live out of the country are assigned a value of –4, valid skip.

COLLAPSED UNEMPLOYMENT RATE

Variable Created: UNEMPRATE-COL

To provide a measure of the economic situation in the respondent's area of residence, the dataset includes a variable indicating the unemployment rate. The round 1 NLSY97 unemployment rate variable was constructed using state and metropolitan area labor force data from the May 1998 publication of *Employment and Earnings* for the month of March 1998. The round 2 data were taken from the May 1999 publication for March 1999, the round 3 data were based on the June 2000 publication for March 2000, and the round 4 data were drawn from the June 2001 edition for March 2001. *Employment and Earnings*, published by the U.S. Department of Labor, Bureau of Labor Statistics, lists the size of the civilian labor force and number of unemployed persons for every state and most metropolitan areas. The variable is created as follows:

1. If the respondent lives in a metropolitan area that is listed in *Employment and Earnings*, then the unemployment rate in the NLSY97 variable is the unemployment rate for that metropolitan area. This rate is calculated by dividing the number of unemployed persons by the number of people in the civilian labor force as reported by BLS.

2. If the respondent does not reside in a metropolitan area listed in *Employment and Earnings*, he or she is assigned a "balance of state" unemployment rate. In these cases, the figures provided for the state and its metropolitan areas are used to compute the unemployment rate for the portion of the state that is not represented in any metropolitan statistical area. (Because the *Employment and Earnings* numbers are based on a different set of MSA codes than the NLSY97 geographic variables, there are a few cases in which NLSY97 metropolitan areas do not match those used in the BLS publication. Researchers who need more exact information should contact BLS or NLS User Services about completing a confidentiality agreement and obtaining the NLSY97 Geocode CD-ROM.)

After the MSA or balance-of-state unemployment rate is calculated for each respondent, the variable for the main file CD-ROM is collapsed into ranges (less than 3.0%, 3.0–5.9%, 6.0–8.9%, 9.0–11.9%, 12.0–14.9%, and 15.0% or higher). This collapsed variable protects the privacy and confidentiality of respondents.

NLSY97 Appendix 5:

Income and Assets Variable Creation

HOUSEHOLD INCOME AND ASSETS

Variables Created: CV_HH_NET_WORTH_Y
CV_INCOME_GROSS_YR
CV_HH_POV_RATIO

Variables Used

Name in Program	Question Name on CD	Name in Program	Question Name on CD
YI_300	YINC-300	YAS3752	YAST-3752
YI_400	YINC-400	YAS3756	YAST-3756
YI_1400	YINC-1400	YAS3780	YAST-3780
YI_1500	YINC-1500	YAS3785	YAST-3785
YI_1600	YINC-1600	YAS3790	YAST-3790
YI_1700	YINC-1700	YAS3810	YAST-3810
YI_1800	YINC-1800	YAS3812	YAST-3812
YI_1900	YINC-1900	YAS3840	YAST-3840
YI_2000	YINC-2000	YAS3850	YAST-3850
YI_2100	YINC-2100	YAS3860	YAST-3860
YI_2200	YINC-2200	YAS3880	YAST-3880
YI_2300	YINC-2300	YAS3882	YAST-3882
YI_2400	YINC-2400	YAS3910	YAST-3910
YI_2500	YINC-2500	YAS3920	YAST-3920
YI_2600	YINC-2600	YAS3922	YAST-3922
YI_2700	YINC-2700	YAS3926	YAST-3926
YI_2900	YINC-2900	YAS3950	YAST-3950
YI_3000	YINC-3000	YAS3960	YAST-3960
YI_3100	YINC-3100	YAS3962	YAST-3962
YI_3300	YINC-3300	YAS3966	YAST-3966
YI_3400	YINC-3400	YAS4010	YAST-4010
YI_3900	YINC-3900	YAS4030	YAST-4030
YI_4000	YINC-4000	YAS4032	YAST-4032
YI_4100	YINC-4100	YAS4036	YAST-4036
YI_4200	YINC-4200	YAS4140	YAST-4140
YI_4300	YINC-4300	YAS4160	YAST-4160
YI_4400	YINC-4400	YAS4162	YAST-4162
YI_4500	YINC-4500	YAS4166	YAST-4166
YI_4600	YINC-4600	YAS4270	YAST-4270
YI_4700	YINC-4700	YAS4290	YAST-4290
YI_4800	YINC-4800	YAS4292	YAST-4292
YI_4900	YINC-4900	YAS4296	YAST-4296
YI_5000	YINC-5000	YAS4400	YAST-4400
YI_5100	YINC-5100	YAS4420	YAST-4420
YI_5200	YINC-5200	YAS4422	YAST-4422
YI_5300	YINC-5300	YAS4426	YAST-4426
YI_5400	YINC-5400	YAS4530	YAST-4530
YI_55002-YI_55007	YINC-5500~000001-~000006	YAS4550	YAST-4550
YI_55001	YINC-5500~000000	YAS4552	YAST-4552
YI_5600	YINC-5600	YAS4556	YAST-4556
YI_5700	YINC-5700	YAS4660	YAST-4660
YI_5800	YINC-5800	YAS4680	YAST-4680
YI_5900	YINC-5900	YAS4682	YAST-4682
YI_6400	YINC-6400	YAS4686	YAST-4686
YI_6500	YINC-6500	YAS4790	YAST-4790
YI_6600	YINC-6600	YAS4810	YAST-4810
YI_6700	YINC-6700	YAS4812	YAST-4812
YI_7000	YINC-7000	YAS4816	YAST-4816
YI_7100	YINC-7100	YAS4840	YAST-4840
YI_7200	YINC-7200	YAS4842	YAST-4842
YI_7300	YINC-7300	YAS4846	YAST-4846
YI_7600	YINC-7600	YAS4870	YAST-4870
YI_7700	YINC-7700	YAS4880	YAST-4880
YI_7800	YINC-7800	YAS4900	YAST-4900
YI_7900	YINC-7900	YAS4902	YAST-4902
YI_8000	YINC-8000	YAS4906	YAST-4906

YI_8100	YINC-8100
YI_8200	YINC-8200
YI_8300	YINC-8300
YI_8600	YINC-8600
YI_8700	YINC-8700
YI_8800	YINC-8800
YI_8900	YINC-8900
YI_9000	YINC-9000
YI_9100	YINC-9100
YI_9200	YINC-9200
YI_9300	YINC-9300
YI_9400	YINC-9400
YI_9500	YINC-9500
YI_9600	YINC-9600
YI_9700	YINC-9700
YI_9800	YINC-9800
YI_9900	YINC-9900
YI_10000	YINC-10000
YI_10100	YINC-10100
YI_10200	YINC-10200
YI_10300	YINC-10300
YI_10400	YINC-10400
YI_10500	YINC-10500
YI_10700	YINC-10700
YI_10800	YINC-10800
I1110001-I1110014	YINC-11100.01-.14
I1160001-I1160013	YINC-11600.01-.13
I1170001-I1170013	YINC-11700.01-.13
YAS50	YAST-050
YAS60	YAST-060
YAS70	YAST-070
YAS200	YAST-200
YAS210	YAST-210
YAS600	YAST-600
YAS900	YAST-900
YAS1400	YAST-1400
YAS1500	YAST-1500
YAS1610	YAST-1610
YAS1860	YAST-1860
YAS1862	YAST-1862
YAS1866	YAST-1866
YAS2520	YAST-2520
YAS2550	YAST-2550
YAS2552	YAST-2552
YAS2556	YAST-2556
YAS2760	YAST-2760
YAS2762	YAST-2762
YAS2766	YAST-2766
YAS3310	YAST-3310
YAS3380	YAST-3380
YAS3382	YAST-3382
YAS3386	YAST-3386
YAS3740	YAST-3740
YAS3750	YAST-3750
YAS5040	YAST-5040
YAS5060	YAST-5060
YAS5062	YAST-5062
A506601-A506620	A5066.01-.20
A507401-A507414	YAST-5074.01-.14
A507417	YAST-5074.17
A507419	YAST-5074.19
A507420	YAST-5074.20
A508001-A508014	YAST-5080.01-.14
A508017	YAST-5080.17
A508019	YAST-5080.19
A508020	YAST-5080.20
A508201-A508217	YAST-5082.01-.17
A508219	YAST-5082.19
A508220	YAST-5082.20
YAS5130	YAST-5130
YAS5150	YAST-5150
YAS5152	YAST-5152
A516401-A516412	YAST-5164.01-.12
A516419	YAST-5164.19
A516420	YAST-5164.20
A517001-A517012	YAST-5170.01-.12
A517019	YAST-5170.19
A517020	YAST-5170.20
A517201-A517205	YAST-5172.01-.05
A517207-A517210	YAST-5172.07-.10
A517212	YAST-5172.12
A517219	YAST-5172.19
A517220	YAST-5172.20
YAS5210	YAST-5210
YAS5220	YAST-5220
YAS5222	YAST-5222
YAS5226	YAST-5226
HIRELY01-HIRELY14	HHI_RELY.01-.14
ID	PUBID
ID	PUBID
YAS1864L, YAS1864U	YAST-1864~000001, ~000002
YAS2554L, YAS2554U	YAST-2554~000001, ~000002
YAS2794L, YAS2794U	YAST-2794~000001, ~000002
YAS3384L, YAS3384U	YAST-3384~000001, ~000002
YAS3754L, YAS3754U	YAST-3754~000001, ~000002
YAS3964L, YAS3964U	YAST-3964~000001, ~000002
YAS4034L, YAS4034U	YAST-4034~000001, ~000002
YAS4164L, YAS4164U	YAST-4164~000001, ~000002
YAS4294L, YAS4294U	YAST-4294~000001, ~000002
YAS4424L, YAS4424U	YAST-4424~000001, ~000002
YAS4554L, YAS4554U	YAST-4554~000001, ~000002
YAS4684L, YAS4684U	YAST-4684~000001, ~000002
YAS4814L, YAS4814U	YAST-4814~000001, ~000002
YAS4844L, YAS4844U	YAST-4844~000001, ~000002
YAS4904L, YAS4904U	YAST-4904~000001, ~000002
YAS5224L, YAS5224U	YAST-5224~000001, ~000002
HHSIZE	CV_HH_SIZE
UNDER18	CV_HH_UNDER18

This program creates the household net worth and gross household income variables. The household net worth variable is an actual number that results from adding the values of all assets and subtracting liabilities of the household. The gross household income variable includes total annual cash receipts before taxes from all sources. The program then creates a ratio comparing the household's total income to federal poverty guidelines based on the number of household residents and the number of members under age 18.

Researchers should note that, like many income and asset variables in the data set, these three variables are topcoded to protect respondent privacy. More information about topcoding is available in the *NLSY97 User's Guide*.

```
/******************* SECTION 1:  GROSS HOUSEHOLD INCOME *********************/

/* In income section, flag top three items involved the most respondents: I. nonfarm employment income -- flag1400
II. father's income -- flag8700  III. mother's income -- flag9200
        flag=0 indicating accurate income reported
        flag=1 indicating estimated income reported */
/*Create the following components of the hh net worth and gross income, such as income from wages and salaries,
        interest, etc. */
/* Income from non-farm employment (nfarmwgY) */
nfarmwgY=-4;
flag1400=0;
if (YI_1400=1 and YI_1700 ne -1 and YI_1700 ne -2 and YI_1700 ne -3) or (YI_1400=-1 and YI_1600=1 and YI_1700
        ne -1 and YI_1700 ne -2 and YI_1700 ne -3) or (YI_1400=-2 and YI_1500=1 and YI_1700 ne -1 and YI_1700
        ne -2 and YI_1700 ne -3) or (YI_1400=-2 and YI_1500=-1 and YI_1600=1 and YI_1700 ne -1 and YI_1700 ne
        -2 and YI_1700 ne -3) then nfarmwgY=YI_1700;
if (YI_1400=1 or (YI_1400=-1 and YI_1600=1) or (YI_1400=-2 and YI_1500=1) or (YI_1400=-2 and YI_1500=-1 and
        YI_1600=1) or (YI_1400=-3 and YI_1600=1) or (YI_1400=-3 and YI_1500=1) or (YI_1400=-2 and YI_1500=-
        3 and YI_1600=1)) and (YI_1700 eq -1 or YI_1700 eq -2 or YI_1700 eq -3) then do;
    if YI_1800=1 then do; nfarmwgY=2500; flag1400=1; end;
    if YI_1800=2 then do; nfarmwgY=7500; flag1400=1; end;
    if YI_1800=3 then do; nfarmwgY=17500; flag1400=1; end;
    if YI_1800=4 then do; nfarmwgY=37500; flag1400=1; end;
    if YI_1800=5 then do; nfarmwgY=75000; flag1400=1; end;
    if YI_1800=6 then do; nfarmwgY=175000; flag1400=1; end;
    if YI_1800=7 then do; nfarmwgY=250001; flag1400=1; end;
  end;
if YI_1400=0 or YI_1500=0 or YI_1600=0 then nfarmwgY=0;
if YI_1600=-1 or YI_1800=-1 then nfarmwgY=-1;
if YI_1500=-2 or YI_1800=-2 then nfarmwgY=-2;
if YI_1600=-3 or YI_1500=-3 or YI_1800=-3 then nfarmwgY=-3;

/* For all the questions below, the youth must be INDEPENDENT (YI_1900=1) */
/* Income from farm and self-employment (farmwgY)*/
farmwgY=-4;
if YI_2000=1 and YI_2100 ne -1 and YI_2100 ne -2 and YI_2100 ne -3 then farmwgY=YI_2100;
if YI_2000=1 and (YI_2100 eq -1 or YI_2100 eq -2 or YI_2100 eq -3) then do;
  if YI_2200=1 then do; farmwgY=-2; end;          if YI_2200=2 then do; farmwgY=2500; end;
  if YI_2200=3 then do; farmwgY=7500; end;        if YI_2200=4 then do; farmwgY=17500; end;
  if YI_2200=5 then do; farmwgY=37500; end;       if YI_2200=6 then do; farmwgY=75000; end;
  if YI_2200=7 then do; farmwgY=175000; end;      if YI_2200=8 then do; farmwgY=250001; end;
end;
if YI_2000=0 then farmwgY=0;
if YI_2000=-1 or YI_2200=-1 then farmwgY=-1;
if YI_2000=-2 or YI_2200=-2 then farmwgY=-2;
if YI_2000=-3 or YI_2200=-3 then farmwgY=-3;

/* Non-farm income from the spouse/partner (nfarmwgP and farmwgP) */
nfarmwgP=-4;
if (YI_2300=1 and YI_2400=1 and YI_2600 ne -1 and YI_2600 ne -2 and YI_2600 ne -3) or (YI_2300=1 and
        YI_2400=-1 and YI_2500=1 and YI_2600 ne -1 and YI_2600 ne -2 and YI_2600 ne -3) then
        nfarmwgP=YI_2600;
if (YI_2300=1 and YI_2400=1 and (YI_2600 eq -1 or YI_2600 eq -2 or YI_2600 eq -3)) or (YI_2300=1 and YI_2400=-1
        and YI_2500=1 and (YI_2600 eq -1 or YI_2600 eq -2 or YI_2600 eq -3)) then do;
    if YI_2700=1 then do; nfarmwgP=2500; end;       if YI_2700=2 then do; nfarmwgP=7500; end;
    if YI_2700=3 then do; nfarmwgP=17500; end;      if YI_2700=4 then do; nfarmwgP=37500; end;
    if YI_2700=5 then do; nfarmwgP=75000; end;      if YI_2700=6 then do; nfarmwgP=175000; end;
    if YI_2700=7 then do; nfarmwgP=250001; end;
```

```
  end;
if YI_2300=0 or YI_2400=0 or YI_2500=0 then nfarmwgP=0;
if YI_2300=-1 or YI_2500=-1 or YI_2700=-1 then nfarmwgP=-1;
if YI_2300=-2 or YI_2400=-2 or YI_2500=-2 or YI_2700=-2 then nfarmwgP=-2;
if YI_2300=-3 or YI_2400=-3 or YI_2500=-3 or YI_2700=-3 then nfarmwgP=-3;

/* Farm-related income from the spouse/partner */
farmwgP=-4;
if YI_2300=1 and YI_2900=1 and YI_3000 ne -1 and YI_3000 ne -2 and YI_3000 ne -3 then farmwgP=YI_3000;
if YI_2300=1 and YI_2900=1 and (YI_3000 eq -1 or YI_3000 eq -2 or YI_3000 eq -3) then do;
  if YI_3100=1 then do; farmwgP=-2; end;          if YI_3100=2 then do; farmwgP=2500; end;
  if YI_3100=3 then do; farmwgP=7500; end;        if YI_3100=4 then do; farmwgP=17500; end;
  if YI_3100=5 then do; farmwgP=37500; end;       if YI_3100=6 then do; farmwgP=75000; end;
  if YI_3100=7 then do; farmwgP=175000; end;      if YI_3100=8 then do; farmwgP=250001; end;
end;
if YI_2300=0 or YI_2900=0 then farmwgP=0;
if YI_2300=-1 or YI_2900=-1 or YI_3100=-1 then farmwgP=-1;
if YI_2300=-2 or YI_2900=-2 or YI_3100=-2 then farmwgP=-2;
if YI_2300=-3 or YI_2900=-3 or YI_3100=-3 then farmwgP=-3;

/* Child support payment (childsuY) */
childsuY=-4;
if YI_3300=1 and YI_3900=1 and YI_4000=1 and YI_4100 ne -1 and YI_4100 ne -2 and YI_4100 ne -3 then
        childsuY=YI_4100;
if YI_3300=1 and YI_3900=1 and YI_4000=1 and (YI_4100 eq -1 or YI_4100 eq -2 or YI_4100 eq -3) then do;
  if YI_4200=1 then do; childsuY=500; end;        if YI_4200=2 then do; childsuY=1750; end;
  if YI_4200=3 then do; childsuY=3750; end;       if YI_4200=4 then do; childsuY=7500; end;
  if YI_4200=5 then do; childsuY=17500; end;      if YI_4200=6 then do; childsuY=37500; end;
  if YI_4200=7 then do; childsuY=50001; end;
end;
if YI_3300=0 or YI_3900=0 or YI_4000=0 then childsuY=0;
if YI_3300=-1 or YI_3900=-1 or YI_4000=-1 or YI_4200=-1 then childsuY=-1;
if YI_3300=-2 or YI_3900=-2 or YI_4000=-2 or YI_4200=-2 then childsuY=-2;
if YI_3300=-1 or YI_3900=-1 or YI_4000=-3 or yI_4100=-3 or YI_4200=-3 then childsuY=-3;

/* Interest payments received by the youth (Y) and his/her partner/spouse */
interesY=-4;
if YI_4300=1 and YI_4400 ne -1 and YI_4400 ne -2 and YI_4400 ne -3 then interesY=YI_4400;
if YI_4300=1 and (YI_4400 eq -1 or YI_4400 eq -2 or YI_4400 eq -3) then do;
  if YI_4500=1 then do; interesY=250; end;        if YI_4500=2 then do; interesY=750; end;
  if YI_4500=3 then do; interesY=1750; end;       if YI_4500=4 then do; interesY=3750; end;
  if YI_4500=5 then do; interesY=6250; end;       if YI_4500=6 then do; interesY=8750; end;
  if YI_4500=7 then do; interesY=10001; end;
end;
if YI_4300=0 then interesy=0;
if YI_4300=-1 or YI_4500=-1 then interesY=-1;
if YI_4300=-2 or YI_4500=-2 then interesY=-2;
if YI_4300=-3 or YI_4400=-3 or YI_4500=-3 then interesY=-3;

/* Dividends from stocks and mutual funds */
dividend=-4;
if YI_4600=1 and YI_4700 ne -1 and YI_4700 ne -2 and YI_4700 ne -3 then dividend=YI_4700;
if YI_4600=1 and (YI_4700 eq -1 or YI_4700 eq -2 or YI_4700 eq -3) then do;
  if YI_4800=1 then do; dividend=250; end;        if YI_4800=2 then do; dividend=750; end;
  if YI_4800=3 then do; dividend=1750; end;       if YI_4800=4 then do; dividend=3750; end;
  if YI_4800=5 then do; dividend=6250; end;       if YI_4800=6 then do; dividend=8750; end;
  if YI_4800=7 then do; dividend=10001; end;
```

```
end;
if YI_4600=0 then dividend=0;
if YI_4600=-1 or YI_4800=-1 then dividend=-1;
if YI_4600=-2 or YI_4800=-2 then dividend=-2;
if YI_4600=-3 or YI_4700=-3 or YI_4800=-3 then dividend=-3;

/* Rental income */
rentalIY=-4;
if YI_4900=1 and YI_5000 ne -1 and YI_5000 ne -2 and YI_5000 ne -3 then rentalIY=YI_5000;
if YI_4900=1 and (YI_5000 eq -1 or YI_5000 eq -2 or YI_5000 eq -3) then do;
  if YI_5100=1 then do; rentalIY=500; end;          if YI_5100=2 then do; rentalIY=1750; end;
  if YI_5100=3 then do; rentalIY=3750; end;         if YI_5100=4 then do; rentalIY=7500; end;
  if YI_5100=5 then do; rentalIY=17500; end;        if YI_5100=6 then do; rentalIY=37500; end;
  if YI_5100=7 then do; rentalIY=50001; end;
end;
if YI_4900=0 then rentalIY=0;
if YI_4900=-1 or YI_5100=-1 then rentalIY=-1;
if YI_4900=-2 or YI_5100=-2 then rentalIY=-2;
if YI_4900=-3 or YI_5000=-3 or YI_5100=-3 then rentalIY=-3;

/* Property or money from estates, trusts, annuities or inheritances */
estatesY=-4;
if YI_5200=1 and YI_5300 ne -1 and YI_5300 ne -2 and YI_5300 ne -3 then estatesY=YI_5300;
if YI_5200=1 and (YI_5300 eq -1 or YI_5300 eq -2 or YI_5300 eq -3) then do;
  if YI_5400=1 then do; estatesY=2500; end;         if YI_5400=2 then do; estatesY=7500; end;
  if YI_5400=3 then do; estatesY=17500; end;        if YI_5400=4 then do; estatesY=37500; end;
  if YI_5400=5 then do; estatesY=75000; end;        if YI_5400=6 then do; estatesY=175000; end;
  if YI_5400=7 then do; estatesY=250001; end;
end;
if YI_5200=0 then estatesY=0;
if YI_5200=-1 or YI_5400=-1 then estatesY=-1;
if YI_5200=-2 or YI_5400=-2 then estatesY=-2;
if YI_5200=-3 or YI_5300=-3 or YI_5400=-3 then estatesY=-3;

/* Income from other sources: SS payments, pension and retirement income, alimony, payments from insurance policies,
        etc... */
pensionY=-4;
if YI_7600=1 and YI_7700 ne -1 and YI_7700 ne -2 and YI_7700 ne -3 then pensionY=YI_7700;
if YI_7600=1 and (YI_7700 eq -1 or YI_7700 eq -2 or YI_7700 eq -3) then do;
  if YI_7800=1 then do; pensionY=500; end;          if YI_7800=2 then do; pensionY=1750; end;
  if YI_7800=3 then do; pensionY=3750; end;         if YI_7800=4 then do; pensionY=7500; end;
  if YI_7800=5 then do; pensionY=17500; end;        if YI_7800=6 then do; pensionY=37500; end;
  if YI_7800=7 then do; pensionY=50001; end;
end;
if YI_7600=0 then pensionY=0;
if YI_7600=-1 or YI_7800=-1 then pensionY=-1;
if YI_7600=-2 or YI_7800=-2 then pensionY=-2;
if YI_7600=-3 or YI_7700=-3 or YI_7800=-3 then pensionY=-3;

/* If the youth lived with the father, the father's income */
flag8700=0;
faincome=-4;
/* identifying the father is in the household */
if (hirely01=4 or hirely02=4 or hirely03=4 or hirely04=4 or hirely05=4 or hirely06=4 or hirely07=4 or hirely08=4 or
        hirely09=4 or hirely10=4 or hirely11=4 or hirely12=4 or hirely13=4 or hirely14=4) then do;
if YI_8700=1 and YI_8800 ne -1 and YI_8800 ne -2 and YI_8800 ne -3 then faincome=YI_8800;
if YI_8700=1 and (YI_8800 eq -1 or YI_8800 eq -2 or YI_8800 eq -3) then do;
```

```
   if YI_8900=1 then do; faincome=2500; flag8700=1; end;
   if YI_8900=2 then do; faincome=7500; flag8700=1; end;
   if YI_8900=3 then do; faincome=17500; flag8700=1; end;
   if YI_8900=4 then do; faincome=37500; flag8700=1; end;
   if YI_8900=5 then do; faincome=75000; flag8700=1; end;
   if YI_8900=6 then do; faincome=175000; flag8700=1; end;
   if YI_8900=7 then do; faincome=250001; flag8700=1; end;
end;
if YI_8700=0 then faincome=0;
if YI_8700=-1 or YI_8900=-1 then faincome=-1;
if YI_8700=-2 or YI_8900=-2 then faincome=-2;
if YI_8700=-3 or YI_8800=-3 or YI_8900=-3 then faincome=-3;
end;

/* If the youth lived with the mother, the mother's income */
flag9200=0;
maincome=-4;
/* identify the mother is in the household */
if (hirely01=3 or hirely02=3 or hirely03=3 or hirely04=3 or hirely05=3 or hirely06=3 or hirely07=3 or hirely08=3 or
          hirely09=3 or hirely10=3 or hirely11=3 or hirely12=3 or hirely13=3 or hirely14=3) then do;
   if YI_9200=1 and YI_9300 ne -1 and YI_9300 ne -2 and YI_9300 ne -3 then maincome=YI_9300;
   if YI_9200=1 and (YI_9300 eq -1 or YI_9300 eq -2 or YI_9300 eq -3) then do;
      if YI_9400=1 then do; maincome=2500; flag9200=1; end;
      if YI_9400=2 then do; maincome=7500; flag9200=1; end;
      if YI_9400=3 then do; maincome=17500; flag9200=1; end;
      if YI_9400=4 then do; maincome=37500; flag9200=1; end;
      if YI_9400=5 then do; maincome=75000; flag9200=1; end;
      if YI_9400=6 then do; maincome=175000; flag9200=1; end;
      if YI_9400=7 then do; maincome=250001; flag9200=1; end;
   end;
   if YI_9200=0 then maincome=0;
   if YI_9200=-1 or YI_9400=-1 then maincome=-1;
   if YI_9200=-2 or YI_9400=-2 then maincome=-2;
   if YI_9200=-3 or YI_9400=-3 then maincome=-3;
end;

/* If the youth lives with the male guardian */
mgincome=-4;
if YI_9600=1 and YI_9700=1 and YI_9800 ne -1 and YI_9800 ne -2 and YI_9800 ne -3 then mgincome=YI_9800;
if YI_9600=1 and YI_9700=1 and (YI_9800 eq -1 or YI_9800 eq -2 or YI_9800 eq -3) then do;
   if YI_9900=1 then do; mgincome=2500; end;          if YI_9900=2 then do; mgincome=7500; end;
   if YI_9900=3 then do; mgincome=17500; end;         if YI_9900=4 then do; mgincome=37500; end;
   if YI_9900=5 then do; mgincome=75000; end;         if YI_9900=6 then do; mgincome=175000; end;
   if YI_9900=7 then do; mgincome=250001;end;
end;
if YI_9700=0 then mgincome=0;
if YI_9600=-1 or YI_9700=-1 or YI_9900=-1 then mgincome=-1;
if YI_9600=-2 or YI_9700=-2 or YI_9900=-2 then mgincome=-2;
if YI_9600=-3 or YI_9700=-3 or YI_9800=-3 or YI_9900=-3 then mgincome=-3;

/* If the youth lives with female guardian */
fgincome=-4;
if YI_10100=1 and YI_10200=1 and YI_10300 ne -1 and YI_10300 ne -2 and YI_10300 ne -3 then
          fgincome=YI_10300;
if YI_10100=1 and YI_10200=1 and (YI_10300 eq -1 or YI_10300 eq -2 or YI_10300 eq -3) then do;
   if YI_10400=1 then do; fgincome=2500; end;         if YI_10400=2 then do; fgincome=7500; end;
   if YI_10400=3 then do; fgincome=17500; end;        if YI_10400=4 then do; fgincome=37500; end;
```

```
  if YI_10400=5 then do; fgincome=75000; end;          if YI_10400=6 then do; fgincome=175000; end;
  if YI_10400=7 then do; fgincome=250001; end;
end;
if YI_10200=0 then fgincome=0;
if YI_10100=-1 or YI_10200=-1 or YI_10400=-1 then fgincome=-1;
if YI_10100=-2 or YI_10200=-2 or YI_10400=-2 then fgincome=-2;
if YI_10100=-3 or YI_10200=-3 or YI_10300=-3 or YI_10400=-3 then fgincome=-3;

/* Income from household members of 14 or older other than spouse/partner */
array otfamI otfamI01-otfamI14;
array I11100 I1110001-I1110014;
array I11600 I1160001-I1160014;
array I11700 I1170001-I1170014;

do I=1 to 14;
  otfamI(I)=-4;
  if YI_10800=1 then do;
    if I11100(I)=0 and I11600(I) ne -1 and I11600(I) ne -2 and I11600(I) ne -3 and I11600(I) ne -4 then
          otfamI(I)=I11600(I);
    if I11100(I)=0 and (I11600(I) eq -1 or I11600(I) eq -2 or I11600(I) eq -3) then do;
      if I11700(I)=1 then do; otfamI(I)=2500; end;          if I11700(I)=2 then do; otfamI(I)=7500; end;
      if I11700(I)=3 then do; otfamI(I)=17500; end;         if I11700(I)=4 then do; otfamI(I)=37500; end;
      if I11700(I)=5 then do; otfamI(I)=75000; end;         if I11700(I)=6 then do; otfamI(I)=175000; end;
      if I11700(I)=7 then do; otfamI(I)=250001; end;
    end;
    if I11100(I)=-1 or I11700(I)=-1 then otfamI(I)=-1;
    if I11100(I)=-2 or I11700(I)=-2 then otfamI(I)=-2;
    if I11100(I)=-3 or I11600(I)=-3 or I11700(I)=-3 then otfamI(I)=-3;
  end;
end;

/* We now create gross hh income according to the youth*/
groshhIY=0;
 if YI_1900=1 then do;
   if nfarmwgY not in (-1,-2,-3,-4) then groshhIY=groshhIY+nfarmwgY;
   if farmwgY not in (-1,-2,-3,-4) then groshhIY=groshhIY+farmwgY;
   if nfarmwgP not in (-1,-2,-3,-4) then groshhIY=groshhIY+nfarmwgP;
   if farmwgP not in (-1,-2,-3,-4) then groshhIY=groshhIY+farmwgP;
   if childsuY not in (-1,-2,-3,-4) then groshhIY=groshhIY+childsuY;
   if interesY not in (-1,-2,-3,-4) then groshhIY=groshhIY+interesY;
   if dividend not in (-1,-2,-3,-4) then groshhIY=groshhIY+dividend;
   if rentalIY not in (-1,-2,-3,-4) then groshhIY=groshhIY+rentalIY;
   if estatesY not in (-1,-2,-3,-4) then groshhIY=groshhIY+estatesY;
   if pensionY not in (-1,-2,-3,-4) then groshhIY=groshhIY+pensionY;
   if faincome not in (-1,-2,-3,-4) then groshhIY=groshhIY+faincome;
   if maincome not in (-1,-2,-3,-4) then groshhIY=groshhIY+maincome;
   if mgincome not in (-1,-2,-3,-4) then groshhIY=groshhIY+mgincome;
   if fgincome not in (-1,-2,-3,-4) then groshhIY=groshhIY+fgincome;
   if prgamt not in (-1,-2,-3,-4) then groshhIY=groshhIY+prgamt;
   do I=1 to 14; if otfamI[I] not in (-1,-2,-3,-4) then groshhIY=groshhIY+otfamI[I]; end;
 end;

 if YI_1900=-1 or YI_8500=-1 or (nfarmwgY=-1 or farmwgY=-1 or nfarmwgP=-1 or farmwgP=-1 or childsuY=-1
          or interesY=-1 or dividend=-1 or rentalIY=-1 or estatesY=-1 or pensionY=-1 or faincome=-1 or maincome=-1
          or mgincome=-1 or fgincome=-1 or otfamI01=-1 or otfamI02=-1 or otfamI03=-1 or otfamI04=-1 or
          otfamI04=-1 or otfamI05=-1 or otfamI06=-1 or otfamI07=-1 or otfamI08=-1 or otfamI09=-1 or
          otfamI10=-1 or otfamI11=-1 or otfamI12=-1 or otfamI13=-1 or otfamI14=-1 or prgamt=-1) then groshhIY=-1;
```

```
if YI_1900=-2 or YI_8500=-2 or (nfarmwgY=-2 or farmwgY=-2 or nfarmwgP=-2 or farmwgP=-2 or childsuY=-2
        or interesY=-2 or dividend=-2 or rentalIY=-2 or pensionY=-2 or estatesY=-2 or faincome=-2 or maincome=-2
        or mgincome=-2 or fgincome=-2 or otfamI01=-2 or otfamI02=-2 or otfamI03=-2 or otfamI04=-2 or
        otfamI04=-2 or otfamI05=-2 or otfamI06=-2 or otfamI07=-2 or otfamI08=-2 or otfamI09=-2 or otfamI10=-2
        or otfamI11=-2 or otfamI12=-2 or otfamI13=-2 or otfamI14=-2 or prgamt=-2) then groshhIY=-2;
if YI_1900=-3 or YI_8500=-3 or (nfarmwgY=-3 or farmwgY=-3 or nfarmwgP=-3 or farmwgP=-3 or childsuY=-3
        or interesY=-3 or dividend=-3 or rentalIY=-3 or pensionY=-3 or estatesY=-3 or faincome=-3 or maincome=-3
        or mgincome=-3 or fgincome=-3 or otfamI01=-3 or otfamI02=-3 or otfamI03=-3 or otfamI04=-3 or
        otfamI04=-3 or otfamI05=-3 or otfamI06=-3 or otfamI07=-3 or otfamI08=-3 or otfamI09=-3 or otfamI10=-3
        or otfamI11=-3 or otfamI12=-3 or otfamI13=-3 or otfamI14=-3 or prgamt=-3) then groshhIY=-3;
if YI_1900=0 or YI_1900=-4  then groshhIY=-4;

if YAS50=-5 then groshhIY=-5; /*give the people who were not interviewed in round 4 value -5.*/
```

/******** SECTION 2: HOUSEHOLD NET WORTH **********/**

```
/* In asset section, flag top three items involved the most respondents: I. savings -- flag4400 II. present value of vehicles
        owned -- flag4790  III. money still owed on these vehicles -- flag4840
        flag=0 indicating accurate income reported
        flag=1 indicating estimated income reported */

/* If the youth OWNS some land, ITS PRESENT VALUE. */
pvranch=0;
if YAS1400=1 then do;
/* If the youth reported owning all of it */
  if YAS1610=1 or YAS1610=2 then do;
    if YAS1860=1 then pvranch=YAS1862;
    if YAS1860=2 then pvranch=YAS1864L+(YAS1864U-YAS1864L)/2;
    if ((YAS1860 ne 1 and YAS1860 ne 2) or (YAS1862=-1 or YAS1862=-2)) then do;
      if YAS1866=1 then do; pvranch=12500; end;        if YAS1866=2 then do; pvranch=37500; end;
      if YAS1866=3 then do; pvranch=75000; end;        if YAS1866=4 then do; pvranch=175000; end;
      if YAS1866=5 then do; pvranch=375000; end;       if YAS1866=6 then do; pvranch=750000; end;
      if YAS1866=7 then do; pvranch=1000001; end;
    end;
  end;
  /* If the youth reported owning only part of it */
  if (YAS1610=3 or YAS1610=4) and YAS2120=100 then do;
    if YAS2140=1 then pvranch=YAS2142;
    if YAS2140=2 then pvranch=(YAS2144L+(YAS2144U-YAS2144L)/2);
    if ((YAS2140 ne 1 and YAS2140 ne 2) or (YAS2142=-1 or YAS2142=-2)) then do;
      if YAS2146=1 then do; pvranch=12500; end;        if YAS2146=2 then do; pvranch=37500; end;
      if YAS2146=3 then do; pvranch=75000; end;        if YAS2146=4 then do; pvranch=175000; end;
      if YAS2146=5 then do; pvranch=375000; end;       if YAS2146=6 then do; pvranch=750000; end;
      if YAS2146=7 then do; pvranch=1000001; end;
    end;
  end;
  if (YAS1610=3 or YAS1610=4) and YAS2120 ne 100 then do;
    if YAS2170=1 then pvranch=YAS2172;
    if YAS2170=2 then pvranch=(YAS2174L+(YAS2174U-YAS2174L)/2);
    if ((YAS2170 ne 1 and YAS2170 ne 2) or (YAS2172=-1 or YAS2172=-2)) then do;
      if YAS2176=1 then do; pvranch=12500; end;        if YAS2176=2 then do; pvranch=37500; end;
      if YAS2176=3 then do; pvranch=75000; end;        if YAS2176=4 then do; pvranch=175000; end;
      if YAS2176=5 then do; pvranch=375000; end;       if YAS2176=6 then do; pvranch=750000; end;
      if YAS2176=7 then do; pvranch=1000001; end;
    end;
  end;
```

```
end;

if YAS1400=-1 or YAS1610=-1 or YAS1864L=-1 or YAS1864U=-1 or YAS1866=-1 or YAS2144L=-1 or YAS2144U=-1
        or YAS2146=-1 or YAS2174L=-1 or YAS2174U=-1 or YAS2176=-1 then pvranch=-1;
if YAS1400=-2 or YAS1610=-2 or YAS1864L=-2 or YAS1864U=-2 or YAS1866=-2 or YAS2144L=-2 or YAS2144U=-2
        or YAS2146=-2 or YAS2174L=-2 or YAS2174U=-2 or YAS2176=-2 then pvranch=-2;
if YAS1400=-3 or YAS1610=-3 or YAS1864L=-3 or YAS1864U=-3 or YAS1866=-3 or YAS2144L=-3 or YAS2144U=-3
        or YAS2146=-3 or YAS2174=-3 or YAS2176=-3 then pvranch=-3;

/* If the youth OWNS the mobile home and the site, ITS PRESENT VALUE.*/
pvmbst=0;
if YAS1500=1 then do;
  if (YAS2520=1 or YAS2520=2) and YAS2550=1 then pvmbst=YAS2552;
  if (YAS2520=1 or YAS2520=2) and YAS2550=2 then pvmbst=YAS2554L+(YAS2554U-YAS2554L)/2;
  if (YAS2520=1 or YAS2520=2) and ((YAS2550 ne 1 and YAS2550 ne 2) or (YAS2552=-1 or YAS2552=-2)) then do;
    if YAS2556=1 then do; pvmbst=500; end;          if YAS2556=2 then do; pvmbst=1750; end;
    if YAS2556=3 then do; pvmbst=3750; end;         if YAS2556=4 then do; pvmbst=7500; end;
    if YAS2556=5 then do; pvmbst=17500; end;        if YAS2556=6 then do; pvmbst=37500; end;
    if YAS2556=7 then do; pvmbst=50001; end;
  end;
end;
if YAS1500=-1 or YAS2520=-1 or YAS2554L=-1 or YAS2554U=-1 or YAS2556=-1 then pvmbst=-1;
if YAS1500=-2 or YAS2520=-2 or YAS2554L=-2 or YAS2554U=-2 or YAS2556=-2 then pvmbst=-2;
if YAS1500=-3 or YAS2520=-3 or YAS2552=-3 or YAS2554L=-3 or YAS2554U=-3 or YAS2556=-3 then pvmbst=-3;

/* If the youth owns only the mobile home, IT'S PRESENT VALUE. */
pvmb=0;
if YAS1500=1 then do;
  if (YAS2520=3 or YAS2520=4) and YAS2760>0 then pvmb=YAS2760;
  if (YAS2520=3 or YAS2520=4) and (YAS2764L>-4 and YAS2764L>-4 and YAS2764L ne .) then
        pvmb=(YAS2764L+(YAS2764U-YAS2764L)/2);
  if (YAS2520=3 or YAS2520=4) and (YAS2760=-1 or YAS2760=-2 or  YAS2960=-3) then do;
    if YAS2766=1 then do; pvmb=12500; end;          if YAS2766=2 then do; pvmb=37500; end;
    if YAS2766=3 then do; pvmb=75000; end;          if YAS2766=4 then do; pvmb=175000; end;
    if YAS2766=5 then do; pvmb=375000; end;         if YAS2766=6 then do; pvmb=750000; end;
    if YAS2766=7 then do; pvmb=1000001; end;
  end;
end;
if YAS1500=-1 or YAS2520=-1 or YAS2764L=-1 or YAS2764U=-1 or YAS2766=-1 then pvmb=-1;
if YAS1500=-2 or YAS2520=-2 or YAS2764L=-2 or YAS2764U=-2 or YAS2766=-2 then pvmb=-2;
if YAS1500=-3 or YAS2520=-3 or YAS2762=-3 or YAS2764L=-3 or YAS2764U=-3 or YAS2766=-3 then pvmb=-3;

/* If the youth owns only the mobile home site, ITS PRESENT VALUE.*/
pvst=0;
if YAS1500=1 then do;
  if (YAS2520=5 or YAS2520=6) and YAS3010=1 then pvst=YAS3012;
  if (YAS2520=5 or YAS2520=6) and YAS3010=2 then pvst=(YAS3014L+(YAS3014U-YAS3014L)/2);
  if (YAS2520=5 or YAS2520=6) and ((YAS3010 ne 1 and YAS3010 ne 2) or (YAS3012=-1 or YAS3012=-2)) then do;
    if YAS3016=1 then do; pvst=12500; end;          if YAS3016=2 then do; pvst=57500; end;
    if YAS3016=3 then do; pvst=75000; end;          if YAS3016=4 then do; pvst=175000; end;
    if YAS3016=5 then do; pvst=575000; end;         if YAS3016=6 then do; pvst=750000; end;
    if YAS3016=7 then do; pvst=1000001; end;
  end;
end;
if YAS2520=-1 or YAS3014L=-1 or YAS3014U=-1 or YAS3016=-1 or YAS1500=-1 then pvst=-1;
if YAS2520=-2 or YAS3014L=-2 or YAS3014U=-2 or YAS3016=-2 or YAS1500=-2 then pvst=-2;
if YAS2520=-3 or YAS3012=-3 or YAS3014L=-3 or YAS3014U=-3 or YAS3016=-3 or YAS1500=-3 then pvst=-3;
```

```
/* If the respondent owns the apartment, its value */
pvapthm=0;
if YAS3310=1 or YAS3310=2 then do;
  if YAS3380=1 then pvapthm=YAS3382;
  if YAS3380=2 then pvapthm=(YAS3384L+(YAS3384U-YAS3384L)/2);
  if ((YAS3380 ne 1 and YAS3380 ne 2) or (YAS3382=-1 or YAS3382=-2)) then do;
    if YAS3386=1 then do; pvapthm=500; end;          if YAS3386=2 then do; pvapthm=1750; end;
    if YAS3386=3 then do; pvapthm=3750; end;         if YAS3386=4 then do; pvapthm=7500; end;
    if YAS3386=5 then do; pvapthm=17500; end;        if YAS3386=6 then do; pvapthm=37500; end;
    if YAS3386=7 then do; pvapthm=50001; end;
  end;
end;
if YAS3310=-1 or YAS3384L=-1 or YAS3384U=-1 or YAS3386=-1 then pvapthm=-1;
if YAS3310=-2 or YAS3384L=-2 or YAS3384U=-2 or YAS3386=-2 then pvapthm=-2;
if YAS3310=-3 or YAS3382=-3 or YAS3384L=-3 or YAS3384U=-3 or YAS3386=-3 then pvapthm=-3;

/* Mortgage or land contract on land or property */
mortgagY=0;
if YAS3740=1 or YAS3740=2 then do;
  if YAS3750=1 and YAS3752 ge 0 then mortgagY=YAS3752;
  if YAS3750=2 and YAS3754U ge 0 and YAS3754L ge 0 then mortgagY=(YAS3754L+(YAS3754U-YAS3754L)/2);
  if ((YAS3750 ne 1 and YAS3750 ne 2) or (YAS3752=-1 or YAS3752=-2)) then do;
    if YAS3756=1 then do; mortgagY=500; end;         if YAS3756=2 then do; mortgagY=1750; end;
    if YAS3756=3 then do; mortgagY=3750; end;        if YAS3756=4 then do; mortgagY=7500; end;
    if YAS3756=5 then do; mortgagY=17500; end;       if YAS3756=6 then do; mortgagY=37500; end;
    if YAS3756=7 then do; mortgagY=50001; end;
  end;
end;
if YAS3740=-1 or YAS3754L=-1 or YAS3754U=-1 or YAS3756=-1 then mortgagY=-1;
if YAS3740=-2 or YAS3754L=-2 or YAS3754U=-2 or YAS3756=-2 then mortgagY=-2;
if YAS3740=-3 or YAS3752=-3 or YAS3754L=-3 or YAS3754U=-3 or YAS3756=-3 then mortgagY=-3;

/* Loans from relatives or friends when respondent bought or remodeled the residence */
loanowed=0;
if YAS3790=1 then do;
  if YAS3810=1 and YAS3812 ge 0 then loanowed=YAS3812;
  if YAS3810=2 and YAS3814U ge 0 and YAS3814L ge 0 then loanowed=(YAS3814L+(YAS3814U-YAS3814L)/2);
  if ((YAS3810 ne 1 and YAS3810 ne 2) or (YAS3812=-1 or YAS3812=-2)) then do;
    if YAS3816=1 then do; loanowed=12500; end;       if YAS3816=2 then do; loanowed=37500; end;
    if YAS3816=3 then do; loanowed=75000; end;       if YAS3816=4 then do; loanowed=175000; end;
    if YAS3816=5 then do; loanowed=375000; end;      if YAS3816=6 then do; loanowed=750000; end;
    if YAS3816=7 then do; loanowed=1000001; end;
  end;
end;
if YAS3780=-1 or YAS3785=-1 or YAS3790=-1 or YAS3814L=-1 or YAS3814U=-1 or YAS3816=-1 then loanowed=-1;
if YAS3780=-2 or YAS3785=-2 or YAS3790=-2 or YAS3814L=-2 or YAS3814U=-2 or YAS3816=-2 then loanowed=-2;
if YAS3780=-3 or YAS3785=-3 or YAS3790=-3 or YAS3812=-3 or YAS3814L=-3 or YAS3814U=-3 or YAS3816=-3 then
        loanowed=-3;

/* Remaining loans from round 3 still unpaid, total amount owed */
stilowed=0;
if (YAS3840=1 and YAS3850=1) or YAS3860=1 then do;
  if YAS3880=1 and YAS3882 ge 0 then stilowed=YAS3882;
  if YAS3880=2 and YAS3884U ge 0 and YAS3884L ge 0 then stilowed=(YAS3884L+(YAS3884U-YAS3884L)/2);
  if ((YAS3880 ne 1 and YAS3880 ne 2) or (YAS3882=-1 or YAS3882=-2)) then do;
    if YAS3886=1 then do; stilowed=500; end;         if YAS3886=2 then do; stilowed=1750; end;
```

```
    if YAS3886=3 then do; stilowed=3750; end;          if YAS3886=4 then do; stilowed=7500; end;
    if YAS3886=5 then do; stilowed=17500; end;         if YAS3886=6 then do; stilowed=37500; end;
    if YAS3886=7 then do; stilowed=50001; end;
  end;
end;
if YAS3860=-1 or YAS3884L=-1 or YAS3884U=-1 or YAS3886=-1 then stilowed=-1;
if YAS3860=-2 or YAS3884L=-2 or YAS3884U=-2 or YAS3886=-2 then stilowed=-2;
if YAS3860=-3 or YAS3882=-3 or YAS3884L=-3 or YAS3884U=-3 or YAS3886=-3 then stilowed=-3;

/* Second mortgages */
secmortY=0;
if YAS3910=1 then do;
  if YAS3920=1 and YAS3922 ge 0 then secmortY=YAS3922;
  if YAS3920=2 and YAS3924U ge 0 and YAS3924L ge 0 then secmortY=(YAS3924L+(YAS3924U-YAS3924L)/2);
  if ((YAS3920 ne 1 and YAS3920 ne 2) or (YAS3922=-1 or YAS3922=-2)) then do;
    if YAS3926=1 then do; secmortY=500; end;           if YAS3926=2 then do; secmortY=1750; end;
    if YAS3926=3 then do; secmortY=3750; end;          if YAS3926=4 then do; secmortY=7500; end;
    if YAS3926=5 then do; secmortY=17500; end;         if YAS3926=6 then do; secmortY=37500; end;
    if YAS3926=7 then do; secmortY=50001; end;
  end;
end;
if YAS3910=-1 or YAS3924L=-1 or YAS3924U=-1 or YAS3926=-1 then secmortY=-1;
if YAS3910=-2 or YAS3924L=-2 or YAS3924U=-2 or YAS3926=-2 then secmortY=-2;
if YAS3910=-3 or YAS3922=-3 or YAS3924L=-3 or YAS3924U=-3 or YAS3926=-3 then secmortY=-3;

/* Taxes on the property to be paid */
proptaxY=0;
if YAS3950=1 then do;
  if YAS3960=1 and YAS3962 ge 0 then proptaxY=YAS3962;
  if YAS3960=2 and YAS3964U ge 0 and YAS3964L ge 0 then proptaxY=(YAS3964L+(YAS3964U-YAS3964L)/2);
  if ((YAS3960 ne 1 and YAS3960 ne 2) or (YAS3962=-1 or YAS3962=-2)) then do;
    if YAS3966=1 then do; proptaxY=500; end;           if YAS3966=2 then do; proptaxY=1750; end;
    if YAS3966=3 then do; proptaxY=3750; end;          if YAS3966=4 then do; proptaxY=7500; end;
    if YAS3966=5 then do; proptaxY=17500; end;         if YAS3966=6 then do; proptaxY=37500; end;
    if YAS3966=7 then do; proptaxY=50001; end;
  end;
end;
if YAS3950=-1 or YAS3964L=-1 or YAS3964U=-1 or YAS3966=-1 then proptaxY=-1;
if YAS3950=-2 or YAS3964L=-2 or YAS3964U=-2 or YAS3966=-2 then proptaxY=-2;
if YAS3950=-3 or YAS3962=-3 or YAS3964L=-3 or YAS3964U=-3 or YAS3966=-3 then proptaxY=-3;

/* Own a business, partnership or professional practice */
pvbuss=0;
if YAS4010=1 or YAS4010=2 then do;
  if YAS4030=1 then pvbuss=YAS4032;
  if YAS4030=2 then pvbuss=(YAS4034L+(YAS4034U-YAS4034L)/2);
  if ((YAS4030 ne 1 and YAS4030 ne 2) or (YAS4032=-1 or YAS4032=-2)) then do;
    if YAS4036=1 then do; pvbuss=12500; end;           if YAS4036=2 then do; pvbuss=37500; end;
    if YAS4036=3 then do; pvbuss=75000; end;           if YAS4036=4 then do; pvbuss=175000; end;
    if YAS4036=5 then do; pvbuss=375000; end;          if YAS4036=6 then do; pvbuss=750000; end;
    if YAS4036=7 then do; pvbuss=1000001; end;
  end;
end;
if YAS4010=-1 or YAS4034L=-1 or YAS4034U=-1 or YAS4036=-1 then pvbuss=-1;
if YAS4010=-2 or YAS4034L=-2 or YAS4034U=-2 or YAS4036=-2 then pvbuss=-2;
if YAS4010=-3 or YAS4032=-3 or YAS4034L=-3 or YAS4034U=-3 or YAS4036=-3 then pvbuss=-3;
```

```
/* Second real estate owned */
secrestY=0;
if YAS4140=1 or YAS4140=2 then do;
  if YAS4160=1 then secrestY=YAS4162;
  if YAS4160=2 then secrestY=(YAS4164L+(YAS4164U-YAS4164L)/2);
  if ((YAS4160 ne 1 and YAS4160 ne 2) or (YAS4162=-1 or YAS4162=-2)) then do;
    if YAS4166=1 then do; secrestY=12500; end;        if YAS4166=2 then do; secrestY=37500; end;
    if YAS4166=3 then do; secrestY=75000; end;        if YAS4166=4 then do; secrestY=175000; end;
    if YAS4166=5 then do; secrestY=375000; end;       if YAS4166=6 then do; secrestY=750000; end;
    if YAS4166=7 then do; secrestY=1000001; end;
  end;
end;
if YAS4140=-1 or YAS4164L=-1 or YAS4164U=-1 or YAS4166=-1 then secrestY=-1;
if YAS4140=-2 or YAS4164L=-2 or YAS4164U=-2 or YAS4166=-2 then secrestY=-2;
if YAS4140=-3 or YAS4162=-3 or YAS4164L=-3 or YAS4164U=-3 or YAS4166=-3 then secrestY=-3;

/* Retirement plans or pensions */
retireY=0;
if YAS4270=1 or YAS4270=2 or YAS4270=3 then do;
  if YAS4290=1 and YAS4292 ge 0 then retireY=YAS4292;
  if YAS4290=2 and YAS4294U ge 0 and YAS4294L ge 0 then retireY=(YAS4294L+(YAS4294U-YAS4294L)/2);
  if ((YAS4290 ne 1 and YAS4290 ne 2) or (YAS4292=-1 or YAS4292=-2)) then do;
    if YAS4296=1 then do; retireY=2500; end;          if YAS4296=2 then do; retireY=7500; end;
    if YAS4296=3 then do; retireY=17500; end;         if YAS4296=4 then do; retireY=37500; end;
    if YAS4296=5 then do; retireY=75000; end;         if YAS4296=6 then do; retireY=175000; end;
    if YAS4296=7 then do; retireY=250001; end;
  end;
end;
if YAS4270=-1 or YAS4294L=-1 or YAS4294U=-1 or YAS4296=-1 then retireY=-1;
if YAS4270=-2 or YAS4294L=-2 or YAS4294U=-2 or YAS4296=-2 then retireY=-2;
if YAS4270=-3 or YAS4292=-3 or YAS4294L=-3 or YAS4294U=-3 or YAS4296=-3 then retireY=-3;

/* Savings in saving accounts, money market, funds, trusts,...*/
savingsY=0;
flag4400=0;
if YAS4400=1 or YAS4400=2 or YAS4400=3 then do;
  if YAS4420=1 and YAS4422 ge 0 then savingsY=YAS4422;
  if YAS4420=2 and YAS4424U ge 0 and YAS4424L ge 0 then savingsY=(YAS4424L+(YAS4424U-YAS4424L)/2);
  if ((YAS4420 ne 1 and YAS4420 ne 2) or (YAS4422=-1 or YAS4422=-2)) then do;
    if YAS4426=1 then do; savingsY=500; flag4400=1; end;
    if YAS4426=2 then do; savingsY=1750; flag4400=1; end;
    if YAS4426=3 then do; savingsY=3750; flag4400=1; end;
    if YAS4426=4 then do; savingsY=7500; flag4400=1; end;
    if YAS4426=5 then do; savingsY=17500; flag4400=1; end;
    if YAS4426=6 then do; savingsY=37500; flag4400=1; end;
    if YAS4426=7 then do; savingsY=50001; flag4400=1; end;
  end;
end;
if YAS4400=-1 or YAS4424L=-1 or YAS4424U=-2 or YAS4426=-1 then savingsY=-1;
if YAS4400=-2 or YAS4424L=-2 or YAS4424U=-2 or YAS4426=-2 then savingsY=-2;
if YAS4400=-3 or YAS4422=-3 or YAS4424L=-3 or YAS4424U=-3 or YAS4426=-3 then savingsY=-3;

/* Other savings in bonds or CDs */
othsavY=0;
if YAS4530=1 or YAS4530=2 or YAS4530=3 then do;
  if YAS4550=1 and YAS4552 ge 0 then othsavY=YAS4552;
  if YAS4550=2 and YAS4554U ge 0 and YAS4554L ge 0 then othsavY=(YAS4554L+(YAS4554U-YAS4554L)/2);
```

```
  if ((YAS4550 ne 1 and YAS4550 ne 2) or (YAS4552=-1 or YAS4552=-2)) then do;
    if YAS4556=1 then do; othsavY=500; end;          if YAS4556=2 then do; othsavY=1750; end;
    if YAS4556=3 then do; othsavY=3750; end;         if YAS4556=4 then do; othsavY=7500; end;
    if YAS4556=5 then do; othsavY=17500; end;        if YAS4556=6 then do; othsavY=37500; end;
    if YAS4556=7 then do; othsavY=50001; end;
  end;
end;
if YAS4530=-1 or YAS4554L=-1 or YAS4554U=-1 or YAS4556=-1 then othsavY=-1;
if YAS4530=-2 or YAS4554L=-2 or YAS4554U=-2 or YAS4556=-2 then othsavY=-2;
if YAS4530=-3 or YAS4552=-3 or YAS4554L=-3 or YAS4554U=-3 or YAS4556=-3 then othsavY=-3;

/* Stocks, mutual funds */
stockY=0;
if YAS4660=1 or YAS4660=2 or YAS4660=3 then do;
  if YAS4680=1 then stockY=YAS4682;
  if YAS4680=2 then stockY=(YAS4684L+(YAS4684U-YAS4684L)/2);
  if ((YAS4680 ne 1 and YAS4680 ne 2) or (YAS4682=-1 or YAS4682=-2)) then do;
    if YAS4686=1 then do; stockY=500; end;           if YAS4686=2 then do; stockY=1750; end;
    if YAS4686=3 then do; stockY=3750; end;          if YAS4686=4 then do; stockY=7500; end;
    if YAS4686=5 then do; stockY=17500; end;         if YAS4686=6 then do; stockY=37500; end;
    if YAS4686=7 then do; stockY=50001; end;
  end;
end;
if YAS4660=-1 or YAS4684L=-1 or YAS4684U=-1 or YAS4686=-1 then stockY=-1;
if YAS4660=-2 or YAS4684L=-2 or YAS4684U=-2 or YAS4686=-2 then stockY=-2;
if YAS4660=-3 or YAS4682=-3 or YAS4684L=-3 or YAS4684U=-3 or YAS4686=-3 then stockY=-3;

/* Present value of vehicles owned */
pvcarsY=0;
flag4790=0;
if YAS4790=1 or YAS4790=2 or YAS4790=3 then do;
  if YAS4810=1 then pvcarsY=YAS4812;
  if YAS4810=2 then pvcarsY=(YAS4814L+(YAS4814U-YAS4814L)/2);
  if ((YAS4810 ne 1 and YAS4810 ne 2) or (YAS4812=-1 or YAS4812=-2)) then do;
    if YAS4816=1 then do; pvcarsY=2500; flag4790=1; end;
    if YAS4816=2 then do; pvcarsY=7500; flag4790=1; end;
    if YAS4816=3 then do; pvcarsY=17500; flag4790=1; end;
    if YAS4816=4 then do; pvcarsY=37500; flag4790=1; end;
    if YAS4816=5 then do; pvcarsY=75000; flag4790=1; end;
    if YAS4816=6 then do; pvcarsY=175000; flag4790=1; end;
    if YAS4816=7 then do; pvcarsY=250001; flag4790=1; end;
  end;
end;
if YAS4790=-1 or YAS4814L=-1 or YAS4814U=-1 or YAS4816=-1 then pvcarsY=-1;
if YAS4790=-2 or YAS4814L=-2 or YAS4814U=-2 or YAS4816=-2 then pvcarsY=-2;
if YAS4790=-3 or YAS4812=-3 or YAS4814L=-3 or YAS4814U=-3 or YAS4376=-3 then pvcarsY=-3;

/* Money still owed on these vehicles */
cardebt=0;
flag4840=0;
if YAS4840=1 then cardebt=YAS4842;
if YAS4840=2 then cardebt=(YAS4844L+(YAS4844U-YAS4844L)/2);
if ((YAS4840 ne 1 and YAS4840 ne 2) or (YAS4842=-1 or YAS4842=-2)) then do;
  if YAS4846=1 then do; cardebt=2500; flag4840=1; end;
  if YAS4846=2 then do; cardebt=7500; flag4840=1; end;
  if YAS4846=3 then do; cardebt=17500; flag4840=1; end;
  if YAS4846=4 then do; cardebt=37500; flag4840=1; end;
```

```
   if YAS4846=5 then do; cardebt=75000; flag4840=1; end;
   if YAS4846=6 then do; cardebt=175000; flag4840=1; end;
   if YAS4846=7 then do; cardebt=250001; flag4840=1; end;
end;
if YAS4790=-1 or YAS4844L=-1 or YAS4844U=-1 or YAS4846=-1 then cardebt=-1;
if YAS4790=-2 or YAS4844L=-2 or YAS4844U=-2 or YAS4846=-2 then cardebt=-2;
if YAS4790=-3 or YAS4842=-3 or YAS4844L=-3 or YAS4844U=-3 or YAS4376=-3 then cardebt=-3;

/* Present value of owned furniture */
pvfurnty=0;
if YAS4870=1 then do; pvfurnty=2500; end;          if YAS4870=2 then do; pvfurnty=7500; end;
if YAS4870=3 then do; pvfurnty=17500; end;         if YAS4870=4 then do; pvfurnty=37500; end;
if YAS4870=5 then do; pvfurnty=75000; end;         if YAS4870=6 then do; pvfurnty=175000; end;
if YAS4870=7 then do; pvfurnty=250001; end;
if YAS4870=-1 then pvfurnty=-1;
if YAS4870=-2 then pvfurnty=-2;
if YAS4870=-3 then pvfurnty=-3;

/* Other assets not being mentioned before */
otassetY=0;
if YAS4880=1 or YAS4880=2 or YAS4880=3 then do;
   if YAS4900=1 then otassetY=YAS4902;
   if YAS4900=2 then otassetY=(YAS4904L+(YAS4904U-YAS4904L)/2);
   if ((YAS4900 ne 1 and YAS4900 ne 2) or (YAS4902=-1 or YAS4902=-2)) then do;
      if YAS4906=1 then do; otassetY=2500; end;          if YAS4906=2 then do; otassetY=7500; end;
      if YAS4906=3 then do; otassetY=17500; end;         if YAS4906=4 then do; otassetY=37500; end;
      if YAS4906=5 then do; otassetY=75000; end;         if YAS4906=6 then do; otassetY=175000; end;
      if YAS4906=7 then do; otassetY=250001; end;
   end;
end;
if YAS4880=-1 or YAS4904L=-1 or YAS4904U=-1 or YAS4906=-1 then otassetY=-1;
if YAS4880=-2 or YAS4904L=-2 or YAS4904U=-2 or YAS4906=-2 then otassetY=-2;
if YAS4880=-3 or YAS4902=-3 or YAS4904L=-3 or YAS4904U=-3 or YAS4376=-3 then otassetY=-3;

/* Loans still owed to family or relatives */
array rloan rloan01 rloan02 rloan03 rloan04 rloan05 rloan06 rloan07 rloan08 rloan09 rloan10 rloan11 rloan12 rloan13
        rloan14 rloan15 rloan16 rloan17 rloan18  rloan19 rloan20;
array A5080 A508001 A508002 A508003 A508004 A508005 A508006 A508007 A508008 A508009 A508010
        A508011 A508012 A508013 A508014 A508015 A508016 A508017 A508018 A508019 A508020;
array A5082 A508201 A508202 A508203 A508204 A508205 A508206 A508207 A508208 A508209 A508210
        A508211 A508212 A508213 A508214 A508215 A508216 A508217 A508218 A508219 A508220;
array A5084L A5084L01 A5084L02 A5084L03 A5084L04 A5084L05 A5084L06 A5084L07 A5084L08 A5084L09
        A5084L10 A5084L11 A5084L12 A5084L13 A5084L14 A5084L15 A5084L16 A5084L17 A5084L18
        A5084L19 A5084L20;
array A5084U A5084U01 A5084U02 A5084U03 A5084U04 A5084U05 A5084U06 A5084U07 A5084U08 A5084U09
        A5084U10 A5084U11 A5084U12 A5084U13 A5084U14 A5084U15 A5084U16 A5084U17 A5084U18
        A5084U19 A5084U20;
array A5086 A508601 A508602 A508603 A508604 A508605 A508606 A508607 A508608 A508609 A508610
        A508611 A508612 A508613 A508614 A508615 A508616 A508617 A508618 A508619 A508620;
array A5170 A517001 A517002 A517003 A517004 A517005 A517006 A517007 A517008 A517009 A517010
        A517011 A517012 A517013 A517014 A517015 A517016 A517017 A517018 A517019 A517020;
array A5172 A517201 A517202 A517203 A517204 A517205 A517206 A517207 A517208 A517209 A517210
        A517211 A517212 A517213 A517214 A517215 A517216 A517217 A517218 A517219 A517220;
array A5174L A5174L01 A5174L02 A5174L03 A5174L04 A5174L05 A5174L06 A5174L07 A5174L08 A5174L09
        A5174L10 A5174L11 A5174L12 A5174L13 A5174L14 A5174L15 A5174L16 A5174L17 A5174L18
        A5174L19 A5174L20;
```

```
array A5174U A5174U01 A5174U02 A5174U03 A5174U04 A5174U05 A5174U06 A5174U07 A5174U08 A5174U09
        A5174U10 A5174U11 A5174U12 A5174U13 A5174U14 A5174U15 A5174U16 A5174U17 A5174U18
        A5174U19 A5174U20;
array A5176 A517601 A517602 A517603 A517604 A517605 A517606 A517607 A517608 A517609 A517610
        A517611 A517612 A517613 A517614 A517615 A517616 A517617 A517618 A517619 A517620;
array A5074 A507401-A507420;
array A5066 A506601-A506620;
array A5164 A516401-A516420;

do I=1 to 20;
  rloan(I)=0;
  if YAS5040=1 and YAS5060 ne 1 and YAS5062 ne 1 then do;
    if A5080(I)=1 then rloan(I)=A5082(I);
    if A5080(I)=2 then rloan(I)=(A5084L(I)+(A5084U(I)-A5084L(I))/2);
    if ((A5080(I) ne 1 and A5080(I) ne 2) or (A5082(I)=-1 or A5082(I)=-2)) then do;
      if A5086(I)=1 then do; rloan(I)=500; end;          if A5086(I)=2 then do; rloan(I)=1750; end;
      if A5086(I)=3 then do; rloan(I)=3750; end;         if A5086(I)=4 then do; rloan(I)=7500; end;
      if A5086(I)=5 then do; rloan(I)=17500; end;        if A5086(I)=6 then do; rloan(I)=37500; end;
      if A5086(I)=7 then do; rloan(I)=50001; end;
    end;
  end;
  if YAS5040=-1 or A5074(I)=-1 or A5086(I)=-1 or A5084L[I]=-1 or A5084U[I]=-1 then rloan(I)=-1;
  if YAS5040=-2 or A5074(I)=-2 or A5086(I)=-2 or A5084L[I]=-2 or A5084U[I]=-2 then rloan(I)=-2;
  if YAS5040=-3 or A5082(I)=-3 or A5086(I)=-3 or A5084L[I]=-3 or A5084U[I]=-3 then rloan(I)=-3;

  if YAS5130=1 and YAS5150 ne 1 and YAS5152 ne 1 then do;
    if A5170(I)=1 then rloan(I)=A5172(I);
    if A5170(I)=2 then rloan(I)=(A5174L(I)+(A5174U(I)-A5174L(I))/2);
    if ((A5170(I) ne 1 and A5170(I) ne 2) or (A5172(I)=-1 or A5172(I)=-2)) then do;
      if A5176(I)=1 then do; rloan(I)=500; end;          if A5176(I)=2 then do; rloan(I)=1750; end;
      if A5176(I)=3 then do; rloan(I)=3750; end;         if A5176(I)=4 then do; rloan(I)=7500; end;
      if A5176(I)=5 then do; rloan(I)=17500; end;        if A5176(I)=6 then do; rloan(I)=37500; end;
      if A5176(I)=7 then do; rloan(I)=50001; end;
    end;
  end;
  if YAS5130=-1 or A5164(I)=-1 or A5176(I)=-1 or A5174L[I]=-1 or A5174U[I]=-1 then rloan(I)=-1;
  if YAS5130=-2 or A5164(I)=-2 or A5176(I)=-2 or A5174L[I]=-2 or A5174U[I]=-2 then rloan(I)=-2;
  if YAS5130=-3 or A5172(I)=-3 or A5176(I)=-3 or A5174L[I]=-3 or A5174U[I]=-3 then rloan(I)=-3;
end;

/* Other debts from loans, credit cards, etc...*/
othdebtY=0;
if YAS5210=1 then do;
  if YAS5220=1 then othdebtY=YAS5222;
  if YAS5220=2 then othdebtY=(YAS5224L+(YAS5224U-YAS5224L)/2);
  if ((YAS5220 ne 1 and YAS5220 ne 2) or (YAS5222=-1 or YAS5222=-2)) then do;
    if YAS5226=1 then do; othdebtY=500; end;          if YAS5226=2 then do; othdebtY=1750; end;
    if YAS5226=3 then do; othdebtY=3750; end;         if YAS5226=4 then do; othdebtY=7500; end;
    if YAS5226=5 then do; othdebtY=17500; end;        if YAS5226=6 then do; othdebtY=37500; end;
    if YAS5226=7 then do; othdebtY=50001; end;
  end;
end;
if YAS5210=-1 or YAS5224L=-1 or YAS5224U=-1 or YAS5226=-1 then othdebtY=-1;
if YAS5210=-2 or YAS5224L=-2 or YAS5224U=-2 or YAS5226=-2 then othdebtY=-2;
if YAS5210=-3 or YAS5222=-3 or YAS5224L=-3 or YAS5224U=-3 or YAS5226=-3 then othdebtY=-3;

/* We now calculate the household net worth according to the youth: hhworthY=assets-liabilities*/
```

```
if YAS50=0 then do;
   hhworthY=0;
   if pvranch not in (-1, -2, -3) then hhworthY=hhworthY+pvranch;
   if pvmbst not in (-1, -2, -3) then hhworthY=hhworthY+pvmbst;
   if pvmb not in (-1, -2, -3) then hhworthY=hhworthY+pvmb;
   if pvst not in (-1, -2, -3) then hhworthY=hhworthY+pvst;
   if pvapthm not in (-1, -2, -3) then hhworthY=hhworthY+pvapthm;
   if pvbuss not in (-1, -2, -3) then hhworthY=hhworthY+pvbuss;
   if secrestY not in (-1, -2, -3) then hhworthY=hhworthY+secrestY;
   if retireY not in (-1, -2, -3) then hhworthY=hhworthY+retireY;
   if savingsY not in (-1, -2, -3) then hhworthY=hhworthY+savingsY;
   if othsavY not in (-1, -2, -3) then hhworthY=hhworthY+othsavY;
   if stockY not in (-1, -2, -3) then hhworthY=hhworthY+stockY;
   if pvcarsY not in (-1, -2, -3) then hhworthY=hhworthY+pvcarsY;
   if pvfurnty not in (-1, -2, -3) then hhworthY=hhworthY+pvfurnty;
   if otassetY not in (-1, -2, -3) then hhworthY=hhworthY+otassetY;
   if mortgagY not in (-1, -2, -3) then hhworthY=hhworthY-mortgagY;
   if loanowed not in (-1, -2, -3) then hhworthY=hhworthY-loanowed;
   if stilowed not in (-1, -2, -3) then hhworthY=hhworthY-stilowed;
   if secmortY not in (-1, -2, -3) then hhworthY=hhworthY-secmortY;
   if cardebt not in (-1, -2, -3) then hhworthY=hhworthY-cardebt;
   if othdebtY not in (-1, -2, -3) then hhworthY=hhworthY-othdebtY;
   do I=1 to 20; if rloan[I] not in (-1, -2, -3) then hhworthY=hhworthY-rloan[I]; end;
end;

if YAS50=1 or YAS50=-4 then hhworthY=-4;

if YAS50=-1 or (pvranch=-1 or pvmbast=-1 or pvmbhm=-1 or pvmbst=-1 or pvapthm=-1 or pvbuss=-1 or secrestY=-1 or
        retireY=-1 or savingsY=-1 or othsavY=-1 or stockY=-1 or pvcarsY=-1 or pvfurnty=-1 or otassetY=-1 or
        mortgagY=-1 or loanowed=-1 or stilowed=-1 or secmortY=-1 or cardebt=-1 or rloan01=-1 or rloan02=-1 or
        rloan03=-1 or rloan04=-1 or rloan05=-1 or rloan06=-1 or rloan07=-1 or rloan08=-1 or rloan09=-1 or rloan10=-
        1 or rloan11=-1 or rloan12=-1 or rloan13=-1 or rloan14=-1 or rloan15=-1 or rloan16=-1 or rloan17=-1 or
        rloan18=-1 or rloan19=-1 or rloan20=-1 or othdebtY=-1) then hhworthY=-1;
if YAS50=-2 or (pvranch=-2 or pvmbast=-2 or pvmbhm=-2 or pvmbst=-2 or pvapthm=-2 or pvbuss=-2 or secrestY=-2 or
        retireY=-2 or savingsY=-2 or othsavY=-2 or stockY=-2 or pvcarsY=-2 or pvfurnty=-2 or otassetY=-2 or
        mortgagY=-2 or loanowed=-2 or stilowed=-2 or secmortY=-2 or cardebt=-2 or rloan01=-2 or rloan02=-2 or
        rloan03=-2 or rloan04=-2 or rloan05=-2 or rloan06=-2 or rloan07=-2 or rloan08=-2 or rloan09=-2 or rloan10=-
        2 or rloan11=-2 or rloan12=-2 or rloan13=-2 or rloan14=-2 or rloan15=-2 or rloan16=-2 or rloan17=-2 or
        rloan18=-2 or rloan19=-2 or rloan20=-2 or othdebtY=-2) then hhworthY=-2;
if YAS50=-3 or (pvranch=-3 or pvmbast=-3 or pvmbhm=-3 or pvmbst=-3 or pvapthm=-3 or pvbuss=-3 or secrestY=-3 or
        retireY=-3 or savingsY=-3 or othsavY=-3 or stockY=-3 or pvcarsY=-3 or pvfurnty=-3 or otassetY=-3 or
        mortgagY=-3 or loanowed=-3 or stilowed=-3 or secmortY=-3 or cardebt=-3 or rloan01=-3 or rloan02=-3 or
        rloan03=-3 or rloan04=-3 or rloan05=-3 or rloan06=-3 or rloan07=-3 or rloan08=-3 or rloan09=-3 or rloan10=-
        3 or rloan11=-3 or rloan12=-3 or rloan13=-3 or rloan14=-3 or rloan15=-3 or rloan16=-3 or rloan17=-3 or
        rloan18=-3 or rloan19=-3 or rloan20=-3 or othdebtY=-3) then hhworthY=-3;
if YAS50=-5 then hhworthY=-5; /*give the people who were not interviewed in round 4 value -5.*/

hhworthY=round(hhworthY,1);
```

/********* SECTION 3: Create the poverty thresholds ************/**

```
povert=-4;

if HHSIZE=-1 or under18=-1 then povert=-1;          if HHSIZE=-2 or under18=-2 then povert=-2;
if HHSIZE=-3 or under18=-3 then povert=-3;

if HHSIZE=1 then povert=8667;
```

```
if HHSIZE=2 then do;
  if under18=0 then povert=11156;          if under18=1 then povert=11483;
  if under18=2 then povert=11483;
end;

if HHSIZE=3 then do;
  if under18=0 then povert=13032;          if under18=1 then povert=13410;
  if under18=2 then povert=13423;          if under18=3 then povert=13423;
end;

if HHSIZE=4 then do;
  if under18=0 then povert=17184;          if under18=1 then povert=17465;
  if under18=2 then povert=16895;          if under18=3 then povert=16954;
  if under18=4 then povert=16954;
end;

if HHSIZE=5 then do;
  if under18=0 then povert=20723;          if under18=1 then povert=21024;
  if under18=2 then povert=20380;          if under18=3 then povert=19882;
  if under18=4 then povert=19578;          if under18=5 then povert=19578;
end;

if HHSIZE=6 then do;
  if under18=0 then povert=23835;          if under18=1 then povert=23930;
  if under18=2 then povert=23436;          if under18=3 then povert=22964;
  if under18=4 then povert=22261;          if under18=5 then povert=21845;
  if under18=6 then povert=21845;
end;

if HHSIZE=7 then do;
  if under18=0 then povert=27425;          if under18=1 then povert=27596;
  if under18=2 then povert=27006;          if under18=3 then povert=26595;
  if under18=4 then povert=25828;          if under18=5 then povert=24934;
  if under18=6 then povert=23953;          if under18=7 then povert=23953;
end;

if HHSIZE=8 then do;
  if under18=0 then povert=30673;          if under18=1 then povert=30944;
  if under18=2 then povert=30387;          if under18=3 then povert=29899;
  if under18=4 then povert=29206;          if under18=5 then povert=28327;
  if under18=6 then povert=27412;          if under18=7 then povert=27180;
  if under18=8 then povert=27180;
end;

if HHSIZE>=9 then do;
  if under18=0 then povert=36897;          if under18=1 then povert=37076;
  if under18=2 then povert=36583;          if under18=3 then povert=36169;
  if under18=4 then povert=35489;          if under18=5 then povert=34554;
  if under18=6 then povert=33708;          if under18=7 then povert=33499;
  if under18=8 then povert=32208;          if under18=9 then povert=32208;
  if under18=10 then povert=32208;         if under18=11 then povert=32208;
end;

povthr=-4;
if groshhIY ge 0 and povert gt 0 then povthr=groshhIY/povert;
if groshhIY eq -1 or povert=-1 then povthr=-1;          if groshhIY eq -2 or povert=-2 then povthr=-2;
```

```
if groshhIY eq -3 or povert=-3 then povthr=-3;          if groshhIY eq -4 then povthr=-4;

povthr2=-4;
if povthr ge 0 then do; povthr2=povthr*100; end;        if povthr eq -1 then povthr2=-1;
if povthr eq -2 then povthr2=-2;                        if povthr eq -3 then povthr2=-3;

povthr3=-4;
if povthr2 ge 0 then do; povthr3=round(povthr2, 1); end;
if povthr2 eq -1 then povthr3=-1;
if povthr2 eq -2 then povthr3=-2;
if povthr2 eq -3 then povthr3=-3;

/*Changing all invalid skips to -3*/
if groshhIY=-1 then groshhIY=-3;     if groshhIY=-2 then groshhIY=-3;
if hhworthY=-1 then hhworthY=-3;  if hhworthY=-2 then hhworthY=-3;
if povthr3=-1 then povthr3=-3;       if povthr3=-2 then povthr3=-3;

/* Repondents who were not interviewed in round 4 */
if YAS50=-5 then povthr3=-5;

                    /*************** SECTION 4:  TopCoding *****************/
if groshhIY ge 138500 then groshhIY=204730;
if povthr3 ge 844 then povthr3=1252;
if hhworthY ge 71448 then hhworthY=147529;

if groshhIY=204730 and povthr3 ne 1252 then do; if povthr3 ge 0 then povthr3=1252; end;
if povthr3=1252 and groshhIY ne 204730 then do; if groshhIY ge 0 then groshhIY=204730; end;

endsas;
```

PARTICIPATION IN GOVERNMENT PROGRAMS

Variables Created: CV_AMT_GOVNT_PGM_PCY.80 – CV_AMT_GOVNT_PGM_PCY.00
CV_GOVNT_PGM_EVER
CV_GOVNT_PGM_YR.80 – CV_GOVNT_PGM_YR.00

Variables Used

Name in Program	Question Name on CD	Name in Program	Question Name on CD
p10	YPRG-010	ps359501-ps359504	YPRG-35950_UPD.01-.04
p1400	YPRG-1400	ps36087	YPRG-36087_UPD
p1210	YPRG-1210	ps36120	YPRG-36120_UPD
ps18301	YPRG-1830_UPD	ps30900A1	YPRG-30900A_UPD.01
ps1897	YPRG-1897_UPD	ps310001m	YPRG-31000_UPD.01~M
ps1930	YPRG-1930_UPD	ps310001y	YPRG-31000_UPD.01~Y
ps40001m, ps40001y	YPRG-4000_UPD.01~M, ~Y	ps310002m, ps310002y	YPRG-31000_UPD.02~M, ~Y
ps40002m, ps40002y	YPRG-4000_UPD.02~M, ~Y	ps316001m, ps316001y	YPRG-31600_UPD.01~M, ~Y
ps42001, ps42002	YPRG-4200_UPD.01, .02	ps316002m, ps316002y	YPRG-31600_UPD.02~M, ~Y
ps43001, ps43002	YPRG-4300_UPD.01, .02	ps318001	YPRG-31800_UPD.01
ps44001, ps44002	YPRG-4400_UPD.01, .02	ps323001, ps323002	YPRG-32300_UPD.01, .02
ps45001, ps45002	YPRG-4500_UPD.01, .02	ps324001	YPRG-32400_UPD.01
ps47001m, ps47001y	YPRG-4700_UPD.01~M, ~Y	ps3250011-ps3240015	YPRG-32500_UPD.01~000001-~000005
ps47002m, ps47002y	YPRG-4700_UPD.02~M, ~Y	ps3250019	YPRG-32500_UPD.01~000999
ps49001, ps49002	YPRG-4900_UPD.01, .02	ps3250021-ps3250025	YPRG-32500_UPD.02~000001-~000005
ps50001, ps50002	YPRG-5000_UPD.01, .02	ps3250029	YPRG-32500_UPD.02~000999
ps52001	YPRG-5200_UPD.01	ps32700A1, ps32700A2	YPRG-32700A_UPD.01, .02
ps59001, ps59002	YPRG-5900_UPD.01, .02	p3500	YPRG-3500
ps60001	YPRG-6000_UPD.01	p40001m, p40001y	YPRG-4000.01~M, ~Y
ps63001, ps63002	YPRG-6300_UPD.01, .02	p42001	YPRG-4200.01
ps98001	YPRG-9800_UPD	p43001	YPRG-4300.01
ps9867	YPRG-9867_UPD	p44001	YPRG-4400.01
ps9900	YPRG-9900_UPD	p47001m, p47001y	YPRG-4700.01~M, ~Y
ps102001m, ps102001y	YPRG-10200_UPD.01~M, ~Y	p49001	YPRG-4900.01
ps104001	YPRG-10400_UPD.01	p50001	YPRG-5000.01
ps105001	YPRG-10500_UPD.01	p52001	YPRG-5200.01
ps106001	YPRG-10600_UPD.01	p59001	YPRG-5900.01
ps109001m, ps109001y	YPRG-10900_UPD.01~M, ~Y	p60001	YPRG-6000.01
ps111001	YPRG-11100_UPD.01	p9700	YPRG-9700
ps112001	YPRG-11200_UPD.01	p102001m, p102001y	YPRG-10200.01~M, ~Y
ps114001	YPRG-11400_UPD.01	p104001	YPRG-10400.01
ps121001	YPRG-12100_UPD.01	p105001	YPRG-10500.01
ps122001	YPRG-12200_UPD.01	p106001	YPRG-10600.01
ps16377	YPRG-16377_UPD	p109001m, p109001y	YPRG-10900.01~M, ~Y
ps16410	YPRG-16410_UPD	p111001	YPRG-11100.01
ps19063A1	YPRG-19063A_UPD.01	p112001	YPRG-11200.01
ps191001m, ps191001y	YPRG-19100_UPD.01~M, ~Y	p114001	YPRG-11400.01
ps191002m, ps191002y	YPRG-19100_UPD.02~M, ~Y	p121001	YPRG-12100.01
ps191003m, ps191003y	YPRG-19100_UPD.03~M, ~Y	p122001	YPRG-12200.01
ps200001m, ps200001y	YPRG-20000_UPD.01~M, ~Y	p18900	YPRG-18900
ps200002m, ps200002y	YPRG-20000_UPD.02~M, ~Y	p194001m, p194001y	YPRG-19400.01~M, ~Y
ps202001	YPRG-20200_UPD.01	p194002m, p194002y	YPRG-19400.02~M, ~Y
ps207001-ps207003	YPRG-20700_UPD.01-.03	p196001, p196002	YPRG-19600.01, .02
ps208001	YPRG-20800_UPD.01	p200001m, p200001y	YPRG-20000.01~M, ~Y
ps2090011-ps2090015	YPRG-20900_UPD.01~000001-~000005	p202001	YPRG-20200.01
ps2090019	YPRG-20900_UPD.01~000999	p207001, p207002	YPRG-20700.01, .02
ps2090021-ps2090025	YPRG-20900_UPD.02~000001-~000005	p208001	YPRG-20800.01
ps2090029	YPRG-20900_UPD.02~000999	p2090011-p2090015	YPRG-20900.01~000001-~000005
ps2090031-ps2090035	YPRG-20900_UPD.03~000001-~000005	p2090021-p2090025	YPRG-20900.02~000001-~000005
ps2090039	YPRG-20900_UPD.03~000999	p21500	YPRG-21500
ps21100A1-ps21100A3	YPRG-21100A_UPD.01-.03	p220001m, p220001y	YPRG-22000.01~M, ~Y
ps21397	YPRG-21397_UPD	p220002m, p220002y	YPRG-22000.02~M, ~Y
ps21430	YPRG-21430_UPD	p222001, p222002	YPRG-22200.01, .02
ps21900A1	YPRG-21900A_UPD.01	p226001m, p226001y	YPRG-22600.01~M, ~Y
ps220001m, ps220001y	YPRG-22000_UPD.01~M, ~Y	p228001	YPRG-22800.01
ps220002m, ps220002y	YPRG-22000_UPD.02~M, ~Y	p233001, p233002	YPRG-23300.01, .02

ps220003m, ps220003y	YPRG-22000_UPD.03~M, ~Y
ps226001m, ps226001y	YPRG-22600_UPD.01~M, ~Y
ps226002m, ps226002y	YPRG-22600_UPD.02~M, ~Y
ps228001-ps228003	YPRG-22800_UPD.01-.03
ps233001-ps233003	YPRG-23300_UPD.01-.03
ps234001, ps234002	YPRG-23400_UPD.01, .02
ps2350011-ps2350015	YPRG-23500_UPD.01~000001, ~000005
ps2350019	YPRG-23500_UPD.01~000999
ps2350021-ps2350025	YPRG-23500_UPD.02~000001-~000005
ps2350029	YPRG-23500_UPD.02~000999
ps2350031-ps2350035	YPRG-23500_UPD.03~000001-~000005
ps2350039	YPRG-23500_UPD.03~000999
ps23700A1-ps23700A3	YPRG-23700A_UPD.01-.03
ps35707	YPRG-35707_UPD
ps35740	YPRG-35740_UPD
ps357901	YPRG-35790_UPD.01
ps358001m, ps358001y	YPRG-35800_UPD.01~M, ~Y
ps358002m, ps358002y	YPRG-35800_UPD.02~M, ~Y
ps358003m, ps358003y	YPRG-35800_UPD.03~M, ~Y
ps358004m, ps358004y	YPRG-35800_UPD.04~M, ~Y
ps358401m, ps358401y	YPRG-35840_UPD.01~M, ~Y
ps358402m, ps358402y	YPRG-35840_UPD.02~M, ~Y
ps358601-ps358604	YPRG-35860_UPD.01-.04
ps359001-ps359004	YPRG-35900_UPD.01-.04
ps359101-ps359104	YPRG-35910_UPD.01-.04
ps3592011-ps3592015	YPRG-35920_UPD.01~000001-~000005
ps3592021-ps3592025	YPRG-35920_UPD.02~000001-~000005
ps3592031-ps3592035	YPRG-35920_UPD.03~000001-~000005
ps3592041-ps3592045	YPRG-35920_UPD.04~000001-~000005
p234001, p234002	YPRG-23400.01, .02
p2350011-p2350015	YPRG-23500.01~000001-~000005
p2350021-p2350023	YPRG-23500.02~000001-~000003
p16200	YPRG-16200
p167001m, p167001y	YPRG-16700.01~M, ~Y
p169001	YPRG-16900.01
p172001m, p172001y	YPRG-17200.01~M, ~Y
p175001	YPRG-17500.01
p181001	YPRG-18100.01
p1830011-p1830015	YPRG-18300.01~000001-~000005
p30500	YPRG-30500
p310001m, p310001y	YPRG-31000.01~M, ~Y
p310002m, p310002y	YPRG-31000.02~M, ~Y
p312001, p312002	YPRG-31200.01, .02
p316001m, p316001y	YPRG-31600.01~M, ~Y
p318001	YPRG-31800.01
p323001, p323002	YPRG-32300.01, .02
p324001	YPRG-32400.01
p3250011-p3250015	YPRG-32500.01~000001-~000005
p3250021- p3250025	YPRG-32500.02~000001-~000005
ID	PUBID
BDATE_D	SYMBOL!KEY!BDATE~D
BDATE_M	SYMBOL!KEY!BDATE~M
BDATE_Y	SYMBOL!KEY!BDATE~Y
YINT_D	SYMBOL!CURDATE~D
YINT_M	SYMBOL!CURDATE~M
YINT_Y	SYMBOL!CURDATE~Y

This program creates several variables describing the respondent's participation in government programs for the economically disadvantaged. During the interview, respondents report amounts received and months of participation in Aid to Families with Dependent Children (AFDC); food stamps; and Women, Infants, and Children (WIC). There is also an "other assistance" question to capture information about any other government program from which respondents may have received assistance. The interview also records amounts received from unemployment compensation and worker's compensation in separate question series; this program also creates variables describing respondents' participation in unemployment or worker's compensation.

The program to create these variables first creates a month-by-month participation array for five of the six categories (AFDC, food stamps, WIC, other programs, and unemployment compensation; the array is not created for worker's compensation). These month-by-month variables constitute part of the event history array for program participation; see appendix 7 for more information. After all five arrays are created, the program merges data to create the summary variables.

Because worker's compensation can be reported either as a lump sum received on a single day or as payments over the period of time that the respondent was not working, the month-by-month array variables are no longer created for this category of program participation. The total benefits received in each calendar year are calculated, but the weeks and months that compensation were received are not created.

```
***to take care of variables that are missing in previous rounds;
array miss

g1total99 g1total00 g1total01 g2total00 g2total01 g3total01
ga1total99 ga1total00 ga1total01 ga2total00 ga2total01 ga3total01

ui1total99 ui1total00 ui1total01 ui2total00 ui2total01 ui3total01
uia1total99 uia1total00 uia1total01
uia2total80-uia2total96 uia2total00 uia2total01
uia3total80-uia3total96
ui3_spells01
```

```
wc1total99 wc1total00 wc1total01 wc2total00 wc2total01 wc3total01
wca1total99 wca1total00 wca1total01 wca2total80 wca2total00 wca2total01 wca3total80-wca3total95 wca3total01;

do over miss; miss=-4; end;

if uia2total99=0 and ui2total99=0 then uia2total99=-4;
if uia3total99=0 and ui3total99=0 then uia3total99=-4;
if uia3total00=0 and ui3total00=0 then uia3total00=-4;

if ga2total99=0 and g2total99=0 then ga2total99=-4;
if ga3total99=0 and g3total99=0 then ga3total99=-4;
if ga3total00=0 and g3total00=0 then ga3total00=-4;

if wca2total99=0 and wc2total99=0 then wca2total99=-4;
if wca3total99=0 and wc3total99=0 then wca3total99=-4;
if wca3total00=0 and wc3total00=0 then wca3total00=-4;

***Hand Edits; ***Currently Receiving information was inconsistent;
if id=1833611 then do; ps1930=0; ps44001=0; end;
if id=1274481 then do; ps1930=1; ps44001=1; end;

***For id=1237111 stop date is missing although was receiving at DLI and not currently receiving. This person denied
        ever having received in Round 5 so no stop date has been imputed;

***Hand Edits for missing months in govt. programs;
if id=1037161 then do; g3total83=9; g3total84=12; g3total85=12; g3total86=12; g3total87=12; g3total88=12;
        g3total89=12; g3total90=12; g3total91=12; g3total92=4; g3prg_ever=109; end;
if id=1602611 then do; g3total89=8; g3total90=12; g3total91=12; g3total92=12; g3total93=12; g3total94=6;
        g3prg_ever=62; end;
if id=1700962 then do; g3total82=6; g3total83=12; g3total84=12; g3total85=12; g3total86=12; g3total87=12;
        g3total88=12; g3total89=12; g3total90=12; g3prg_ever=210; end;

***change weekly ui figures to daily amount;
if PS59001 gt 0 then ps59001=ps59001/7;        if PS59002 gt 0 then ps59002=ps59002/7;
if PS59003 gt 0 then ps59003=ps59003/7;        if PS59004 gt 0 then ps59004=ps59004/7;
if P59001 gt 0 then p59001=p59001/7;           if P59002 gt 0 then p59002=p59002/7;
if P59003 gt 0 then p59003=p59003/7;           if P59004 gt 0 then p59003=p59004/7;

***change weekly wc figures to daily amount;
if PS121001 gt 0 then ps121001=ps121001/7;     if PS121002 gt 0 then ps121002=ps121002/7;
if PS121003 gt 0 then ps121003=ps121003/7;     if PS121004 gt 0 then ps121004=ps121004/7;
if P121001 gt 0 then p121001=p121001/7;        if P121002 gt 0 then p121002=p121002/7;
if P121003 gt 0 then p121003=p121003/7;        if P121004 gt 0 then p121004=p121004/7;

***change previous round variables to account for noninterviews;
array gtotal     (i) gtotal80-gtotal99 gtotal00 gtotal01;
array g3total    (i) g3total80-g3total99 g3total00 g3total01;
array g2total    (i) g2total80-g2total99 g2total00 g2total01;
array g1total    (i) g1total80-g1total99 g1total00 g1total01;
array gatotal    (i) gatotal80-gatotal99 gatotal00 gatotal01;
array ga3total   (i) ga3total80-ga3total99 ga3total00 ga3total01;
array ga2total   (i) ga2total80-ga2total99 ga2total00 ga2total01;
array ga1total   (i) ga1total80-ga1total99 ga1total00 ga1total01;

array uitotal    (i) uitotal80-uitotal99 uitotal00 uitotal01;
array ui3total   (i) ui3total80-ui3total99 ui3total00 ui3total01;
array ui2total   (i) ui2total80-ui2total99 ui2total00 ui2total01;
```

```
array ui1total   (i) ui1total80-ui1total99 ui1total00 ui1total01;
array uiatotal   (i) uiatotal80-uiatotal99 uiatotal00 uiatotal01;
array uia3total  (i) uia3total80-uia3total99 uia3total00 uia3total01;
array uia2total  (i) uia2total80-uia2total99 uia2total00 uia2total01;
array uia1total  (i) uia1total80-uia1total99 uia1total00 uia1total01;
array ui_spells  (i) ui_spells80-ui_spells99 ui_spells00 ui_spells01;
array ui3_spells (i) ui3_spells80-ui3_spells99 ui3_spells00 ui3_spells01;
array ui2_spells (i) ui2_spells80-ui2_spells99 ui2_spells00 ui2_spells01;
array ui1_spells (i) ui1_spells80-ui1_spells99 ui1_spells00 ui1_spells01;

array wctotal    (i) wctotal80-wctotal99 wctotal00 wctotal01;
array wc3total   (i) wc3total80-wc3total99 wc3total00 wc3total01;
array wc2total   (i) wc2total80-wc2total99 wc2total00 wc2total01;
array wc1total   (i) wc1total80-wc1total99 wc1total00 wc1total01;
array wcatotal   (i) wcatotal80-wcatotal99 wcatotal00 wcatotal01;
array wca3total  (i) wca3total80-wca3total99 wca3total00 wca3total01;
array wca2total  (i) wca2total80-wca2total99 wca2total00 wca2total01;
array wca1total  (i) wca1total80-wca1total99 wca1total00 wca1total01;
array nu         (i) nu80-nu99 nu00 nu01;

do i=1 to 22; nu=0; end;

*** if r3 interview, use r3 created var;          *** if noninterview in r3, then use r2 variable;
*** if noninterview in r2, then use r1 variable; *** if variable not in r1, r2 or r3 then set to 0;
do i=1 to 22;
  gtotal=g3total;
  if g3total=-5 then do;
    if g2total=-5 then do; gtotal=g1total; end;
    if g2total ne -5 then do; gtotal=g2total; end;
  end;
  if gtotal=. then gtotal=0;

  gatotal=ga3total;
  if ga3total=-5 then do;
    if ga2total=-5 then do; gatotal=ga1total; end;
    if ga2total ne -5 then do; gatotal=ga2total; end;
  end;
  if gatotal=. then gatotal=0;

  uitotal=ui3total;
  if ui3total=-5 then do;
    if ui2total=-5 then do; uitotal=ui1total; end;
    if ui2total ne -5 then do; uitotal=ui2total; end;
  end;
  if uitotal=. then uitotal=0;

  uiatotal=uia3total;
  if uia3total=-5 then do;
    if uia2total=-5 then do; uiatotal=uia1total; end;
    if uia2total ne -5 then do; uiatotal=uia2total; end;
  end;
  if uiatotal=. then uiatotal=0;

  ui_spells=ui3_spells;
  if ui3_spells=-5 then do;
    if ui2_spells=-5 then do; ui_spells=ui1_spells; end;
    if ui2_spells ne -5 then do; ui_spells=ui2_spells; end;
```

```
   end;
   if ui_spells=. then ui_spells=0;

   wctotal=wc3total;
   if wc3total=-5 then do;
      if wc2total=-5 then do; wctotal=wc1total; end;
      if wc2total ne -5 then do; wctotal=wc2total; end;
   end;
   if wctotal=. then wctotal=0;

   wcatotal=wca3total;
   if wca3total=-5 then do;
      if wca2total=-5 then do; wcatotal=wca1total; end;
      if wca2total ne -5 then do; wcatotal=wca2total; end;
   end;
   if wcatotal=. then wcatotal=0;
end;

wc_ever=wc3_ever;
if wc3_ever=-5 then do;
   if wc2_ever ne -5 then do; wc_ever=wc2_ever; end;
   if wc2_ever=-5 then do; wc_ever=wc1_ever; end;
end;
if wc_ever=-4 then do; wc_ever=0; end;

ui_ever=ui3_ever;
if ui3_ever=-5 then do;
   if ui2_ever ne -5 then do; ui_ever=ui2_ever; end;
   if ui2_ever=-5 then do; ui_ever=ui1_ever; end;
end;
if ui_ever=-4 then do; ui_ever=0; end;

gprg_ever=g3prg_ever;
if g3prg_ever=-5 then do;
   if g2prg_ever ne -5 then do; gprg_ever=g2prg_ever; end;
   if g2prg_ever=-5 then do; gprg_ever=g1prg_ever; end;
end;
if gprg_ever=-4 then do; gprg_ever=0; end;

ack=0;   ackm=0;     ackh=0;   fck=0;     fckm=0;   fckh=0;
wck=0;   wckm=0;     wckh=0;   ock=0;     ockm=0;   ockh=0;
uick=0;  uickt=0;    uickm=0;  wcck=0;    wcckt=0;  wcckm=0;

cm14=((bdate_y+14)-1980)*12+bdate_m;          cmb=((bdate_y)-1980)*12+bdate_m;
ym14=(BDATE_Y*100+BDATE_M)+1400;              iym=YINT_Y*100+YINT_M;

if (p1210=1 and int99_y=-5) then do;                /*not interviewed in round 3 but interviewed in round 2*/
   if int98_y ne -5 then do;
      dliym=int98_y*100+(int98_m+1); dlicm=(int98_y-1980)*12+(int98_m+1);
      li_day=int98_d; li_mo=int98_m; li_year=int98_y;
   end;
   if int98_y=-5 then do;                           /*not interviewed round 3 or 2*/
      dliym=int97_y*100+(int97_m+1); dlicm=(int97_y-1980)*12+(int97_m+1);
      li_day=int97_d; li_mo=int97_m; li_year=int97_y;
   end;
end;
```

```
if (p1210=1 and int99_y>-5) then do;                    /*interviewed in round 3*/
   dliym=int99_y*100+(int99_m+1); dlicm=(int99_y-1980)*12+(int99_m+1);
   li_day=int99_d; li_mo=int99_m; li_year=int99_y;
end;
if p1210 ne 1 then dliym=ym14;      if p1210 ne 1 then dlicm=cm14;

doicm=(YINT_Y-1980)*12+YINT_M;
aaicm=(YINT_Y-BDATE_Y)*12+YINT_M-BDATE_M;

array UI    (l)   UI001-UI258;              array uiamt (l)   uiamt001-uiamt258;
array WC    (l)   WC001-WC258;              array wcamt (l)   wcamt001-wcamt258;
array F     (l)   F001-F258;                array famt  (l)   famt001-famt258;
array fhhm  (l)   fhhm001-fhhm258;          array W     (l)   W001-W258;
array wamt  (l)   wamt001-wamt258;          array whhm  (l)   whhm001-whhm258;
array A     (l)   A001-A258;                array aamt  (l)   aamt001-aamt258;
array ahhm  (l)   ahhm001-ahhm258;          array O     (l)   O001-O258;
array oamt  (l)   oamt001-oamt258;          array ohhm  (l)   ohhm001-ohhm258;

/*2. [b1] begin - Rs eligible for a program in round 3*/
if P1210>-4 then do;
  if PS35707>-3 then P16200=PS35707;            /**AFDC**/
  if PS16377>-3 then P18900=PS16377;            /**Food Stamps**/
  if PS36087>-3 then P30500=PS36087;            /**Other**/
  if PS1897>-3 then P3500=PS1897;               /**Unemployment Compensation**/
  if PS21397>-3 then P21500=PS21397;            /**WIC**/
  if PS9867>-3 then P9700=PS9867;               /**Worker's Compensation**/

  do l=1 to 258;
    if cmb le L le doicm then do; A=0; W=0; F=0; O=0; WC=0; UI=0; end;
  end;

  /** start YEAR information **/
  array ysu1 (j)   PS40001Y   PS40002Y   PS40003Y   PS40004Y;        /**unemployment, sdli**/
  array ysu2 (j)   P40001Y    P40002Y    P40003Y    P40004Y;         /**unemployment, int**/
  array ysc1 (j)   PS102001Y  PS102002Y  PS102003Y  PS102004Y;       /**workers comp, sdil**/
  array ysc2 (j)   P102001Y   P102002Y   P102003Y   P102004Y;        /**workers comp, int**/
  array ysf1 (j)   PS191001Y  PS191002Y  PS191003Y  PS191004Y;       /**food stamps, sdli**/
  array ysf2 (j)   P194001Y   P194002Y   P194003Y   P194004Y;        /**food stamps, int**/
  array ysw1 (j)   PS220001Y  PS220002Y  PS220003Y  PS220004Y;       /**wic, sdli**/
  array ysw2 (j)   P220001Y   P220002Y   P220003Y   P220004Y;        /**wic, int**/
  array ysa1 (j)   PS358001Y  PS358002Y  PS358003Y  PS358004Y;       /**afdc, sdli**/
  array ysa2 (j)   P167001Y   P167002Y   P167003Y   P167004Y;        /**afdc, int**/
  array yso1 (j)   PS310001Y  PS310002Y  PS310003Y  PS310004Y;       /**other, sdli**/
  array yso2 (j)   P310001Y   P310002Y   P310003Y   P310004Y;        /**other, int**/

  /**  start MONTH information **/
  array msu1 (j)   PS40001M   PS40002M   PS40003M   PS40004M;        /**unemployment, sdli**/
  array msu2 (j)   P40001M    P40002M    P40003M    P40004M;         /**unemployment, int**/
  array msc1 (j)   PS102001M  PS102002M  PS102003M  PS102004M;       /**workers comp, sdil**/
  array msc2 (j)   P102001M   P102002M   P102003M   P102004M;        /**workers comp, int**/
  array msf1 (j)   PS191001M  PS191002M  PS191003M  PS191004M;       /**food stamps, sdli**/
  array msf2 (j)   P194001M   P194002M   P194003M   P194004M;        /**food stamps, int**/
  array msw1 (j)   PS220001M  PS220002M  PS220003M  PS220004M;       /**wic, sdli**/
  array msw2 (j)   P220001M   P220002M   P220003M   P220004M;        /**wic, int**/
  array msa1 (j)   PS358001M  PS358002M  PS358003M  PS358004M;       /**afdc, sdli**/
  array msa2 (j)   P167001M   P167002M   P167003M   P167004M;        /**afdc, int**/
  array mso1 (j)   PS310001M  PS310002M  PS310003M  PS310004M;       /**other, sdli**/
```

```
array mso2 (j)   P310001M   P310002M   P310003M   P310004M;      /**other, int**/

/**start DAY information for wc or ui only **/
array dsu1 (j)   PS42001    PS42002    PS42003    PS42004;     /**unemployment, sdli**/
array esu1 (j)   PS43001    PS43002    PS43003    PS43004;     /**estimated unemployment, sdli**/
array dsu2 (j)   P42001     P42002     P42003     P42004;      /**unemployment, int**/
array esu2 (j)   P43001     P43002     P43003     P43004;      /**estimated unemployment, sdli**/
array dsc1 (j)   PS104001   PS104002   PS104003   PS104004;/**workers comp, sdil**/
array esc1 (j)   PS105001   PS105002   PS105003   PS105004;/**estimated workers comp, sdil**/
array dsc2 (j)   P104001    P104002    P104003    P104004;     /**workers comp, int**/
array esc2 (j)   P105001    P105002    P105003    P105004;     /**estimated workers comp, sdil**/

/** end YEAR information **/
array yeu1 (j)   PS47001Y   PS47002Y   PS47003Y   PS47004Y;          /**unemployment, sdli**/
array yeu2 (j)   P47001Y    P47002Y    P47003Y    P47004Y;           /**unemployment, int**/
array yec1 (j)   PS109001Y  PS109002Y  PS109003Y  PS109004Y;         /**workers comp, sdil**/
array yec2 (j)   P109001Y   P109002Y   P109003Y   P109004Y;          /**workers comp, int**/
array yef1 (j)   PS200001Y  PS200002Y  PS200003Y  PS200004Y;         /**food stamps, sdli**/
array yef2 (j)   P200001Y   P200002Y   P200003Y   P200004Y;          /**food stamps, int**/
array yew1 (j)   PS226001Y  PS226002Y  PS226003Y  PS226004Y;         /**wic, sdli**/
array yew2 (j)   P226001Y   P226002Y   P226003Y   P226004Y;          /**wic, int**/
array yea1 (j)   PS358401Y  PS358402Y  PS358403Y  PS358404Y;         /**afdc, sdli**/
array yea2 (j)   P172001Y   P172002Y   P172003Y   P172004Y;          /**afdc, int**/
array yeo1 (j)   PS316001Y  PS316002Y  PS316003Y  PS316004Y;         /**other, sdli**/
array yeo2 (j)   P316001Y   P316002Y   P316003Y   P316004Y;          /**other, int**/

/** end MONTH information **/
array meu1 (j)   PS47001M   PS47002M   PS47003M   PS47004M;      /**unemployment, sdli**/
array meu2 (j)   P47001M    P47002M    P47003M    P47004M;       /**unemployment, int**/
array mec1 (j)   PS109001M  PS109002M  PS109003M  PS109004M;  /**workers comp, sdil**/
array mec2 (j)   P109001M   P109002M   P109003M   P109004M;      /**workers comp, int**/
array mef1 (j)   PS200001M  PS200002M  PS200003M  PS200004M;  /**food stamps, sdli**/
array mef2 (j)   P200001M   P200002M   P200003M   P200004M;      /**food stamps, int**/
array mew1 (j)   PS226001M  PS226002M  PS226003M  PS226004M;/**wic, sdli**/
array mew2 (j)   P226001M   P226002M   P226003M   P226004M;      /**wic, int**/
array mea1 (j)   PS358401M  PS358402M  PS358403M  PS358404M;  /**afdc, sdli**/
array mea2 (j)   P172001M   P172002M   P172003M   P172004M;      /**afdc, int**/
array meo1 (j)   PS316001M  PS316002M  PS316003M  PS316004M;  /**other, sdli**/
array meo2 (j)   P316001M   P316002M   P316003M   P316004M;      /**other, int**/

/**end DAY information for wc or ui only **/
array deu1  (j)   PS49001    PS49002    PS49003   PS49004;          /**unemployment, sdli**/
array eeu1  (j)   PS50001    PS50002    PS50003   PS50004;          /**estimated unemployment, sdli**/
array deu2  (j)   P49001     P49002     P49003    P49004;           /**unemployment, int**/
array eeu2  (j)   P50001     P50002     P50003    P50004;           /**estimated unemployment, sdli**/
array dec1  (j)   PS111001   PS111002   PS111003  PS111004;         /**workers comp, sdil**/
array eec1  (j)   PS112001   PS112002   PS112003  PS112004;         /**estimated workers comp, sdil**/
array dec2  (j)   P111001    P111002    P111003   P111004;          /**workers comp, int**/
array eec2  (j)   P112001    P112002    P112003   P112004;          /**estimated workers comp, sdil**/

/**************** A small note about the following arrays**************************
The elements in some arrays have consecutive suffixes signifying the position in the array like in
    array ymsf1 (j) yms10 yms11 ymsf13 ymsf14.
However others do not have consecutive suffixes like in
    array ymsw1 (j)  yms05-yms07 ymsw14 or array ymsf1 (j)  yms10 yms11 ymsf13 ymsf14
```

This means that we do not know how the first two elements had been indexed. So for the rest of the elements we just take the array name and add the number which shows the position of this element in that row. So ymsf13 and ymsf14 are the 3rd and 4th elements of the array named ymsf1. This is done in many of the arrays below.
**********/

```
array ymsu1 (j)  ymsu11-ymsu14;                array ymsu2 (j)  ymsu21-ymsu24;
array ymsc1 (j)  ymsc11-ymsc14;                array ymsc2 (j)  ymsc21-ymsc24;
array ymsf1 (j)  yms10 yms11 ymsf13 ymsf14;    array ymsf2 (j)  yms12-yms14 ymsf24;
array ymsw1 (j)  yms05-yms07 ymsw14;           array ymsw2 (j)  yms08 yms09 ymsw23 ymsw24;
array ymsa1 (j)  yms01-yms03 ymsa14;           array ymsa2 (j)  yms04 ymsa22 ymsa23 ymsa4;
array ymso1 (j)  yms15 yms16 ymso13 ymso14;    array ymso2 (j)  yms17 yms18 ymso23 ymso14;

array csmu1 (j)  csmu11-csmu14;                array csmu2 (j)  csmu21-csmu24;
array csmc1 (j)  csmc11-csmc14;                array csmc2 (j)  csmc21-csmc24;
array csmf1 (j)  csm10 csm11 csmf13 csmf14;    array csmf2 (j)  csm12-csm14 csmf24;
array csmw1 (j)  csm05-csm07 csmw14;           array csmw2 (j)  csm08 csm09 csmw23 csmw24;
array csma1 (j)  csm01-csm03 csma14;           array csma2 (j)  csm04 csma22 csma23 csma24;
array csmo1 (j)  csm15 csm16 csmo13 csmo14;    array csmo2 (j)  csm17 csm18 csmo23 csmo24;

array ymeu1 (j) ymeu11-ymeu14;                 array ymeu2 (j) ymeu21-ymeu24;
array ymec1 (j) ymec11-ymec14;                 array ymec2 (j) ymec21-ymec24;
array ymef1 (j) yme10 yme11 ymef13 ymef14;     array ymef2 (j) yme12-yme14 ymef24;
array ymew1 (j) yme05-yme07 ymew14;            array ymew2 (j) yme08 yme09 ymew23 ymew24;
array ymea1 (j) yme01-yme03 ymea14;            array ymea2 (j) yme04 ymea22 ymea23 ymea24;
array ymeo1 (j) yme15 yme16 ymeo13 ymeo14;     array ymeo2 (j) yme17 yme18 ymeo23 ymeo24;

array cemu1 (j)  cemu11-cemu14;                array cemu2 (j)  cemu21-cemu24;
array cemc1 (j)  cemc11-cemc14;                array cemc2 (j)  cemc21-cemc24;
array cemf1 (j)  cem10 cem11 cemf13 cemf14;    array cemf2 (j)  cem12-cem14 cemf24;
array cemw1 (j)  cem05-cem07 cemw14;           array cemw2 (j)  cem08 cem09 cemw23 cemw24;
array cema1 (j)  cem01-cem03 cema14;           array cema2 (j)  cem04 cema22 cema23 cema24;
array cemo1 (j)  cem15 cem16 cemo13 cemo14;    array cemo2 (j)  cem17 cem18 cemo23 cemo24;

array su1d  (j) su11_d su12_d su13_d su14_d;   array eu1d  (j) eu11_d eu12_d eu13_d eu14_d;
array su2d  (j) su21_d su22_d su23_d su24_d;   array eu2d  (j) eu21_d eu22_d eu23_d eu24_d;
array sw1d  (j) sw11_d sw12_d sw13_d sw14_d;   array ew1d  (j) ew11_d ew12_d ew13_d ew14_d;
array sw2d  (j) sw21_d sw22_d sw23_d sw24_d;   array ew2d  (j) ew21_d ew22_d ew23_d ew24_d;

/** CURRENTLY receiving information **/
array curu1 (j)  PS44001    PS44002    PS44003    PS44004;          /**unemployment, sdli**/
array curu2 (j)  P44001     P44002     P44003     P44004;           /**unemployment, int**/
array curc1 (j)  PS106001   PS106002   PS106003   PS106004;         /**workers comp, sdil**/
array curc2 (j)  P106001    P106002    P106003    P106004;          /**workers comp, int**/
array curf1 (j)  PS21100A1  PS21100A2  PS21100A3  PS21100A4;        /**food stamps, no gap sdli**/
array curf2 (j)  P196001    P196002    P196003    P196004;          /**food stamps, int**/
array curw1 (j)  PS23700A1  PS23700A2  PS23700A3  PS23700A4;        /**wic, no gap sdli**/
array curw2 (j)  P222001    P222002    P222003    P222004;          /**wic, int**/
array cura1 (j)  PS359501   PS359502   PS359503   PS359504;         /**afdc, no gap sdli**/
array cura2 (j)  P169001    P169002    P169003    P169004;          /**afdc, int**/
array curo1 (j)  PS32700A1  PS32700A2  PS32700A3  PS32700A4;        /**other, no gap sdli**/
array curo2 (j)  P312001    P312002    P312003    P312004;          /**other, int**/

/** EDIT FLAGS for dates **/
array eflagu1 (j)  eflagu11-eflagu14;          /**unemployment, sdli**/
array eflagu2 (j)  eflagu21-eflagu24;          /**unemployment, int**/
array eflagc1 (j)  eflagc11-eflagc14;          /**workers comp, sdil**/
array eflagc2 (j)  eflagc21-eflagc24;          /**workers comp, int**/
```

```
 array eflagf1 (j)  eflag10 eflag11 eflagf13 eflagf14;          /**food stamps, sdli**/
 array eflagf2 (j)  eflag12-eflag14 eflagf24;                   /**food stamps, int**/
 array eflagw1 (j)  eflag05-eflag07 eflagw14;                   /**wic, sdli**/
 array eflagw2 (j)  eflag08 eflag09 eflagw23 eflagw24;          /**wic, int**/
 array eflaga1 (j)  eflag01-eflag03 eflaga14;                   /**afdc, sdli**/
 array eflaga2 (j)  eflag04 eflaga22 eflaga23 eflaga24;         /**afdc, int**/
 array eflago1 (j)  eflag15 eflag16 eflago13 eflago14;          /**other, sdli**/
 array eflago2 (j)  eflag17 eflag18 eflago23 eflago24;          /**other, int**/

 /** EDIT FLAGS for amounts **/
 array aflagu1 (j)  aflagu11-aflagu14;                          /**unemployment, sdli**/
 array aflagu2 (j)  aflagu21-aflagu24;                          /**unemployment, int**/
 array aflagc1 (j)  aflagc11-aflagc14;                          /**workers comp, sdil**/
 array aflagc2 (j)  aflagc21-aflagc24;                          /**workers comp, int**/
 array aflagf1 (j)  aflag10 aflag11 aflagf13 aflagf14;          /**food stamps, sdli**/
 array aflagf2 (j)  aflag12-aflag14 aflagf24;                   /**food stamps, int**/
 array aflagw1 (j)  aflag05-aflag07 aflagw14;                   /**wic, sdli**/
 array aflagw2 (j)  aflag08 aflag09 aflagw23 aflagw24;          /**wic, int**/
 array aflaga1 (j)  aflag01-aflag03 aflaga14;                   /**afdc, sdli**/
 array aflaga2 (j)  aflag04 aflaga22 aflaga23 aflaga24;         /**afdc, int**/
 array aflago1 (j)  aflag15 aflag16 aflago13 aflago14;          /**other, sdli**/
 array aflago2 (j)  aflag17 aflag18 aflago23 aflago24;          /**other, int**/

/** AMOUNT RECEIVED - actual & estimated **/
 array wincu1  (j)  PS59001    PS59002    PS59003   PS59004;  /**unemployment, amount sdli**/
 array wincu2  (j)  P59001     P59002     P59003    P59004;   /**unemployment, amount int**/
 array ewincu1 (j)  PS60001    PS60002    PS60003   PS60004;  /**unemployment, est amount sdli**/
 array ewincu2 (j)  P60001     P60002     P60003    P60004;   /**unemployment, est amount int**/

 array wincc1  (j)  PS121001   PS121002   PS121003  PS121004;  /**workers comp, amount sdli**/
 array wincc2  (j)  P121001    P121002    P121003   P121004;   /**workers comp, amount int**/
 array ewincc1 (j)  PS122001   PS122002   PS122003  PS122004;  /**workers comp, est amount sdli**/
 array ewincc2 (j)  P122001    P122002    P122003   P122004;   /**workers comp, est amount int**/

 array incf1   (j)  PS207001   PS207002   PS207003  PS207004;  /**food stamps, amount sdli**/
 array incf2   (j)  P207001    P207002    P207003   P207004;   /**food stamps, amount int**/
 array eincf1  (j)  PS208001   PS208002   PS208003  PS208004;  /**food stamps, est amount sdli**/
 array eincf2  (j)  P208001    P208002    P208003   P208004;   /**food stamps, est amount int**/

 array incw1   (j)  PS233001   PS233002   PS233003  PS233004;  /**wic, amount sdli**/
 array incw2   (j)  P233001    P233002    P233003   P233004;   /**wic, amount int**/
 array eincw1  (j)  PS234001   PS234002   PS234003  PS234004;  /**wic, est amount sdli**/
 array eincw2  (j)  P234001    P234002    P234003   P234004;   /**wic, est amount int**/

 array inca1   (j)  PS359001   PS359002   PS359003  PS359004;  /**afdc, amount sdli**/
 array inca2   (j)  P181001    P181002    P181003   P181004;   /**afdc, amount int**/
 array einca1  (j)  PS359101   PS359102   PS359103  PS359104;  /**afdc, est amount sdli**/
 array einca2  (j)  P182001    P182002    P182003   P182004;   /**afdc, est amount int**/

 array inco1   (j)  PS323001   PS323002   PS323003  PS323004;  /**other, amount sdli**/
 array inco2   (j)  P323001    P323002    P323003   P323004;   /**other, amount int**/
 array einco1  (j)  PS324001   PS324002   PS324003  PS324004;  /**other, est amount sdli**/
 array einco2  (j)  P324001    P324002    P324003   P324004;   /**other, est amount int**/

 /** ESTIMATED WEEKS - unable to answer start or stop quex**/
 array wksu1   (j)  PS52001    PS52002    PS52003   PS52004;   /**unemployment, est weeks sdli**/
 array wksu2   (j)  P52001     P52002     P52003    P52004;    /**unemployment, est weeks int**/
```

```
array wksc1   (j)  PS114001   PS114002   PS114003  PS114004;        /**workers comp, est weeks sdli**/
array wksc2   (j)  P114001    P114002    P114003   P114004;         /**workers comp, est weeks int**/

array wksf1   (j)  PS202001   PS202002   PS202003  PS202004;        /**food stamps, est weeks sdli**/
array wksf2   (j)  P202001    P202002    P202003   P202004;         /**food stamps, est weeks int**/

array wksw1   (j)  PS228001   PS228002   PS228003  PS228004;        /**wic, est weeks sdli**/
array wksw2   (j)  P228001    P228002    P228003   P228004;         /**wic, est weeks int**/

array wksa1   (j)  PS358601   PS358602   PS358603  PS358604;        /**afdc, est weeks sdli**/
array wksa2   (j)  P175001    P175002    P175003   P175004;         /**afdc, est weeks int**/

array wkso1   (j)  PS318001   PS318002   PS318003  PS318004;        /**other, est weeks sdli**/
array wkso2   (j)  P318001    P318002    P318003   P318004;         /**other, est weeks int**/


array mosu1   (j)  mosu11-mosu14;        array mosu2   (j)  mosu21-mosu24;
array mosc1   (j)  mosc11-mosc14;        array mosc2   (j)  mosc21-mosc24;
array mosf1   (j)  mosf11-mosf14;        array mosf2   (j)  mosf21-mosf24;
array mosw1   (j)  mosw11-mosw14;        array mosw2   (j)  mosw21-mosw24;
array mosa1   (j)  mosa11-mosa14;        array mosa2   (j)  mosa21-mosa24;
array moso1   (j)  moso11-moso14;        array moso2   (j)  moso21-moso24;

/** PERSON receiving amount, respondent **/
array rcvrf1  (j)  PS2090011  PS2090021  PS2090031  PS2090041;    /**food stamps, sdli**/
array rcvrf2  (j)  P2090011   P2090021   P2090031   P2090041;     /**food stamps, int**/
array rcvrw1  (j)  PS2350011  PS2350021  PS2350031  PS2350041;    /**wic, sdli**/
array rcvrw2  (j)  P2350011   P2350021   P2350031   P2350041;     /**wic, int**/
array rcvra1  (j)  PS3592011  PS3592021  PS3592031  PS3592041;    /**afdc, sdli**/
array rcvra2  (j)  P1830011   P1830021   P1830031   P1830041;     /**afdc, int**/
array rcvro1  (j)  PS3250011  PS3250021  PS3250031  PS3250041;    /**other, sdli**/
array rcvro2  (j)  P3250011   P3250021   P3250031   P3250041;     /**other, int**/

/** PERSON receiving amount, spouse **/
array rcvsf1  (j)  PS2090012  PS2090022  PS2090032  PS2090042;    /**food stamps, sdli**/
array rcvsf2  (j)  P2090012   P2090022   P2090032   P2090042;     /**food stamps, int**/
array rcvsw1  (j)  PS2350012  PS2350022  PS2350032  PS2350042;    /**wic, sdli**/
array rcvsw2  (j)  P2350012   P2350022   P2350032   P2350042;     /**wic, int**/
array rcvsa1  (j)  PS3592012  PS3592022  PS3592032  PS3592042;    /**afdc, sdli**/
array rcvsa2  (j)  P1830012   P1830022   P1830032   P1830042;     /**afdc, int**/
array rcvso1  (j)  PS3250012  PS3250022  PS3250032  PS3250042;    /**other, sdli**/
array rcvso2  (j)  P3250012   P3250022   P3250032   P3250042;     /**other, int**/

/** PERSON receiving amount, child **/
array rcvcf1  (j)  PS2090013  PS2090023  PS2090033  PS2090043;    /**food stamps, sdli**/
array rcvcf2  (j)  P2090013   P2090023   P2090033   P2090043;     /**food stamps, int**/
array rcvcw1  (j)  PS2350013  PS2350023  PS2350033  PS2350043;    /**wic, sdli**/
array rcvcw2  (j)  P2350013   P2350023   P2350033   P2350043;     /**wic, int**/
array rcvca1  (j)  PS3592013  PS3592023  PS3592033  PS3592043;    /**afdc, sdli**/
array rcvca2  (j)  P1830013   P1830023   P1830033   P1830043;     /**afdc, int**/
array rcvco1  (j)  PS3250013  PS3250023  PS3250033  PS3250043;    /**other, sdli**/
array rcvco2  (j)  P3250013   P3250023   P3250033   P3250043;     /**other, int**/

/** PERSON receiving amount family member **/
array rcvff1  (j)  PS2090014  PS2090024  PS2090034  PS2090044;    /**food stamps, sdli**/
array rcvff2  (j)  P2090014   P2090024   P2090034   P2090044;     /**food stamps, int**/
```

```
array rcvfw1  (j)  PS2350014  PS2350024  PS2350034  PS2350044;     /**wic, sdli**/
array rcvfw2  (j)  P2350014   P2350024   P2350034   P2350044;      /**wic, int**/
array rcvfa1  (j)  PS3592014  PS3592024  PS3592034  PS3592044;     /**afdc, sdli**/
array rcvfa2  (j)  P1830014   P1830024   P1830034   P1830044;      /**afdc, int**/
array rcvfo1  (j)  PS3250014  PS3250024  PS3250034  PS3250044;     /**other, sdli**/
array rcvfo2  (j)  P3250014   P3250024   P3250034   P3250044;      /**other, int**/

/** PERSON receiving amount, other person **/
array rcvof1  (j)  PS2090015  PS2090025  PS2090035  PS2090045;     /**food stamps, sdli**/
array rcvof2  (j)  P2090015   P2090025   P2090035   P2090045;      /**food stamps, int**/
array rcvow1  (j)  PS2350015  PS2350025  PS2350035  PS2350045;     /**wic, sdli**/
array rcvow2  (j)  P2350015   P2350025   P2350035   P2350045;      /**wic, int**/
array rcvoa1  (j)  PS3592015  PS3592025  PS3592035  PS3592045;     /**afdc, sdli**/
array rcvoa2  (j)  P1830015   P1830025   P1830035   P1830045;      /**afdc, int**/
array rcvoo1  (j)  PS3250015  PS3250025  PS3250035  PS3250045;     /**other, sdli**/
array rcvoo2  (j)  P3250015   P3250025   P3250035   P3250045;      /**other, int**/

array whof1   (j)  who10 who11 whof13 whof14;
array whow1   (j)  who05-who07 whow14;
array whoa1   (j)  who01-who03 whoa14;
array whoo1   (j)  who15 who16 whoo13 whoo14;

array whof2   (j)  who12-who14 whof24;
array whow2   (j)  who08 who09 whow23 whow24;
array whoa2   (j)  who04 whoa22 whoa23 whoa24;
array whoo2   (j)  who17 who18 whoo23 whoo24;

/** PERSON was receiving at DLI, chk **/
array dlif1   (j)  PS19063A1   PS19063A2   PS19063A3   PS19063A4;  /**food stamps**/
array dliw1   (j)  PS21900A1   PS21900A2   PS21900A3   PS21900A4;  /**wic**/
array dlia1   (j)  PS357901    PS357902    PS357903    PS357904;   /**afdc**/
array dlio1   (j)  PS30900A1   PS30900A2   PS30900A3   PS30900A4;  /**other**/
array dliui1  (j)  PS18301     PS18302     PS18303     PS18304;    /**unemployment**/
array dliwc1  (j)  PS98001     PS98002     PS98003     PS98004;    /**workers comp**/

*** these arrays encompass all programs for the arrays;
array ys    (k)  ysa1 ysa2 ysw1 ysw2 ysf1 ysf2 yso1 yso2 ysu1 ysu2 ysc1 ysc2;
array ms    (k)  msa1 msa2 msw1 msw2 msf1 msf2 mso1 mso2 msu1 msu2 msc1 msc2;
array ye    (k)  yea1 yea2 yew1 yew2 yef1 yef2 yeo1 yeo2 yeu1 yeu2 yec1 yec2;
array me    (k)  mea1 mea2 mew1 mew2 mef1 mef2 meo1 meo2 meu1 meu2 mec1 mec2;
array cur   (k)  cura1 cura2 curw1 curw2 curf1 curf2 curo1 curo2 curu1 curu2 curc1 curc2;
array eflag (k) eflaga1 eflaga2 eflagw1 eflagw2 eflagf1 eflagf2 eflago1 eflago2 eflagu1 eflagu2 eflagc1 eflagc2;
array aflag (k) aflaga1 aflaga2 aflagw1 aflagw2 aflagf1 aflagf2 aflago1 aflago2 aflagu1 aflagu2 aflagc1 aflagc2;
array yms   (k)  ymsa1 ymsa2 ymsw1 ymsw2 ymsf1 ymsf2 ymso1 ymso2 ymsu1 ymsu2 ymsc1 ymsc2;
array wks   (k)  wksa1 wksa2 wksw1 wksw2 wksf1 wksf2 wkso1 wkso2 wksu1 wksu2 wksc1 wksc2;
array cwks  (k)  ps35740 p16200 ps21430 p21500 ps16410 p18900 ps36120 p30500 ps1930 p3500 ps9900 p9700;
array yme   (k)  ymea1 ymea2 ymew1 ymew2 ymef1 ymef2 ymeo1 ymeo2 ymeu1 ymeu2 ymec1 ymec2;
array mos   (k)  mosa1 mosa2 mosw1 mosw2 mosf1 mosf2 moso1 moso2 mosu1 mosu2 mosc1 mosc2;
array csm   (k)  csma1 csma2 csmw1 csmw2 csmf1 csmf2 csmo1 csmo2 csmu1 csmu2 csmc1 csmc2;
array cem   (k)  cema1 cema2 cemw1 cemw2 cemf1 cemf2 cemo1 cemo2 cemu1 cemu2 cemc1 cemc2;
array rcvr  (k)  rcvra1 rcvra2 rcvrw1 rcvrw2 rcvrf1 rcvrf2 rcvro1 rcvro2 rcvr09-rcvr12;
array rcvs  (k)  rcvsa1 rcvsa2 rcvsw1 rcvsw2 rcvsf1 rcvsf2 rcvso1 rcvso2 rcvs09-rcvs12;
array rcvc  (k)  rcvca1 rcvca2 rcvcw1 rcvcw2 rcvcf1 rcvcf2 rcvco1 rcvco2 rcvc09-rcvc12;
array rcvf  (k)  rcvfa1 rcvfa2 rcvfw1 rcvfw2 rcvff1 rcvff2 rcvfo1 rcvfo2 rcvf09-rcvf12;
array rcvo  (k)  rcvoa1 rcvoa2 rcvow1 rcvow2 rcvof1 rcvof2 rcvoo1 rcvoo2 rcvo09-rcvo12;
array who   (k)  whoa1  whoa2  whow1  whow2  whof1  whof2  whoo1  whoo2 who09-who12;
array dli   (k)  dlia1  dlia2  dliw1  dliw2  dlif1  dlif2  dlio1  dlio2 dliui1 dliui2 dliwc1 dliwc2;
array inc   (k)  inca1 inca2 incw1 incw2 incf1 incf2 inco1 inco2 wincu1 wincu2 wincc1 wincc2;

array agefla1 (j) agefla11 agefla12 agefla13 agefla14;
array ageflw1 (j) ageflw11 ageflw12 ageflw13 ageflw14;
array ageflf1 (j) ageflf11 ageflf12 ageflf13 ageflf14;

array agefla2 (j) agefla21 agefla22 agefla23 agefla24;
array ageflw2 (j) ageflw21 ageflw22 ageflw23 ageflw24;
array ageflf2 (j) ageflf21 ageflf22 ageflf23 ageflf24;
```

```
 array ageflo1 (j) ageflo11 ageflo12 ageflo13 ageflo14;          array ageflo2 (j) ageflo21 ageflo22 ageflo23 ageflo24;
 array ageflu1 (j) ageflu11 ageflu12 ageflu13 ageflu14;          array ageflu2 (j) ageflu21 ageflu22 ageflu23 ageflu24;
 array ageflc1 (j) ageflc11 ageflc12 ageflc13 ageflc14;          array ageflc2 (j) ageflc21 ageflc22 ageflc23 ageflc24;
 array agefl (k)  agefla1 agefla2 ageflw1 ageflw2 ageflf1 ageflf2 ageflo1 ageflo2 ageflu1 ageflu2 ageflc1 ageflc2;

 ***arrays to determine days in month that need to be accounted for in wc and ui arrays;
 array st_year (z) ysu1 ysu2 ysc1 ysc2;                  array st_mo   (z) msu1 msu2 msc1 msc2;
 array st_day  (z) dsu1 dsu2 dsc1 dsc2;                  array est_day (z) esu1 esu2 esc1 esc2;
 array en_year (z) yeu1 yeu2 yec1 yec2;                  array en_mo   (z) meu1 meu2 mec1 mec2;
 array en_day  (z) deu1 deu2 dec1 dec2;                  array een_day (z) eeu1 eeu2 eec1 eec2;
 array s_day   (z) su1d su2d sw1d sw2d;                  array e_day   (z) eu1d eu2d ew1d ew2d;
 array dli_day (z) dliui1 dliui2 dliwc1 dliwc2;          array cur_day (z) curu1 curu2 curc1 curc2;
 array est_wks (z) wksu1 wksu2 wksc1 wksc2;

***to determine the length of a spell of unemployment or workers compensation determine days;
do z=1 to 4;
 do j=1 to 4;
   if st_day>0 or est_day>0 then do;                          /*set start day, if given*/
     if st_mo ne en_mo or st_year ne en_year then do;
       if st_mo=1 or st_mo=3 or st_mo=5 or st_mo=7 or st_mo=8 or st_mo=10 or st_mo=12 then do;
         if st_day>0 and st_day ne . then s_day=(32-st_day);
         if est_day=1 then s_day=31; if est_day=2 then s_day=21; if est_day=3 then s_day=11;
       end;
       if st_mo=4 or st_mo=6 or st_mo=9 or st_mo=11 then do;
         if st_day>0 and st_day ne . then s_day=(31-st_day);
         if est_day=1 then s_day=30; if est_day=2 then s_day=20; if est_day=3 then s_day=10;
       end;
       if st_mo=2 then do;
         if st_day>0 and st_day ne . then s_day=(29-st_day);
         if est_day=1 then s_day=28; if est_day=2 then s_day=18; if est_day=3 then s_day=8;
       end;
     end;
   end;
   if st_year>-4 and st_year<0 then do;                  /*start date not given*/
     if dli_day=1 and est_wks>4 then s_day=(32-li_day);
     if dli_day=1 and est_wks>0 and est_wks<5 then s_day=(est_wks*7);
     if dli_day ne 1 and est_wks>4 then s_day=1;
     if dli_day ne 1 and est_wks>0 and est_wks<5 then s_day=(est_wks*7);
   end;
   if st_year=-4 and dli_day=1 then do;
     if (li_mo=1 or li_mo=3 or li_mo=5 or li_mo=7 or li_mo=8 or li_mo=10 or li_mo=12) then do; s_day=32-li_day;
         end;
     if (li_mo=4 or li_mo=6 or li_mo=9 or li_mo=11) then do; s_day=31-li_day; end;
     if li_mo=2 then do; s_day=29-li_day; end;
   end;

   if en_day>0 and en_day ne . then do;                  /*set end day, if given*/
     e_day=en_day;
   end;
   if en_day<0 and een_day>0 and een_day ne . then do; /*end day unknown, estimate given*/
     if een_day=1 then e_day=9; if een_day=2 then e_day=19; if een_day=3 then e_day=28;
   end;
   if cur_day=1 then do; e_day=yint_d; end;
                                   /*start and end month and year are the same*/
   if (st_mo=en_mo and st_year=en_year and 0<st_year and st_year ne . and 0<st_mo and st_mo ne . ) then do;
     if st_day>0 then do; s_day=(e_day-st_day+1); e_day=0; end;
     if -4<st_day<0 and est_day>0 then do;
```

```
        if est_day=1 then s_day=e_day;          if est_day=2 then s_day=e_day-10;
        if est_day=3 then s_day=e_day-20;       e_day=0;
      end;
      if s_day<0 then s_day=0;
    end;
    if ((st_year>-4 and st_year<0) or (en_year>-4 and en_year<0)) and cur_day ne 1 then do; e_day=31; end;
  end;
end;

*************************************HAND EDITS for Special Cases****************************************;
***When start and end mo&yr are the same s_day and e_day are calculated in a particular manneras shown above.
        However when "Currenltly Receving" holds the current interview year is assigned as the end year in the
        following part of the program. So when start mo&yr and current interview mo&yr the same the s_day and
        e_day are not calculated as they should be. These hand edits will take care of that;
if id=1044502 then do; su21_d=1; eu21_d=0; end;
if id=1247931 then do; su11_d=8; eu11_d=0; end;
if id=1518051 then do; su21_d=1; eu21_d=0; end;
if id=1702241 then do; su11_d=2; eu11_d=0; end;
if id=1766431 then do; su11_d=28; eu11_d=0; end;
if id=1880762 then do; su11_d=2; eu11_d=0; end;

*DLI was Nov 4, 1999 and stopped receiving on Dec 15, 1999. So the 27 days of Nov 1999 were not counted in R3
        Program Participation variable. So 27 days of Nov99 and 15 days of Dec99 makes 42 days in the year 1999.;
if id=1322971 then do; su11_d=42; eu11_d=0; end;

*1st Loop began in August 2000 (Begining of the Month) and ended the same month (End of the Month). The second
        Loop began on January 01, 2001 and is still receiving on the date of the current interview which is January 24,
        2001. However since "Currently Receiving" holds for both loops, both eu11_d and eu12_d will be 24.;
if id=1615981 then do; su11_d=28; eu11_d=0; su12_d=24; eu12_d=0; end;

*Ist Loop began in Feb 2000 and ended in the middle of March 2000. This person claims to be currently receiving as of
        the current interview date January 18, 2001. However no start date for the 2nd Loop is known. So we are
        ignoring the 2nd Loop. Since "Currently Receiving" holds eu11_d=18 when it should be 19;
if id=1768281 then do; ew11_d=19; end;

*Gave start year as 1998 with bad month and received for 3 weeks. So start and end month and year were imputed to be
        January 1998. So we do the following hand edit.;
if id=1110561 then do; sw21_d=21; ew21_d=0; end;

/**     This portion of the SAS program defines the start date and end dates.  If the respondent      **
**      reports still receiving, the interview date is used as the temporary end date for the last     **
**      loop reported.  In the next survey round, the respondent will be asked if he or she is still  **
**      recieving and, if not, a permanent end date equivalent to the interview date of the previous  **
**      round will be assigned.  Users will be able to tell which method was used by looking at the   **
**      following participation flag variable created during the program.  The categories are the     **
**      following:                                                                                    **
**              1=respondent reported participation dates                                             **
**              2=start month imputed                                                                 **
**              3=start month and year imputed                                                        **
**              4=stop month imputed                                                                  **
**              5=stop month and year imputed                                                         **
**              6=start and stop dates imputed                                                        **
**              7=error in data due to round 2 programming error                                      **/

  do k=1 to 12;                                 /*[b2]*/
    do j=1 to 4;                                /*[b3]*/
      if ys>0 and ms>0 then yms=(ys*100)+ms;    /**set start date if both start month and year are valid**/
```

```
    if dli=1 then yms=dliym;                    /**set start date to date last int if receiving then**/
    if cur=1 then yme=iym;                      /**set end date to current int date if receiving now**/
    if ye>0 and me>0 then yme=(ye*100)+me;      /**set end date if both end month and year are valid**/
    if yme>iym then yme=iym;                    /**if end month greater than int month, constrain by int month*/
    if yms>0 and yme ge dliym then eflag=1;
    if wks ge 0 and (ys<0 or ms<0 or ye<0 or me<0) then do;
      mos=floor(wks/4.3);                       /**determine months from estimated weeks variable**/
    end;

agefl=0;
if ys>0 and ys<1990 then agefl=1;
```

/**1. if start year is known and month is unknown ****/**

```
/**  1.a' If weeks are known and currently receiving, then count backwards by the number of weeks from the interview
          date. If the number of weeks falls short of the start year, the start month is December of that year. If the number
          of weeks is past the start year, then the start month is January of that year.****/
   if ys>0 and -3 le ms le -1 then do;
    if (wks>0 and cwks=1) then do;
      yme=iym;
      if mos le YINT_M then yms=iym-mos;
      if YINT_M le mos le (YINT_M+12) then yms=iym-100-(mos-12);
      if (YINT_M+12) le mos le (YINT_M+24) then yms=iym-200-(mos-24);
      if (YINT_M+24) le mos le (YINT_M+36) then yms=iym-300-(mos-36);
      if (YINT_M+36) le mos le (YINT_M+48) then yms=iym-400-(mos-48);
      if (YINT_M+48) le mos le (YINT_M+60) then yms=iym-500-(mos-60);
      if (YINT_M+60) le mos le (YINT_M+72) then yms=iym-600-(mos-72);
      if (YINT_M+72) le mos le (YINT_M+84) then yms=iym-700-(mos-84);
      if (YINT_M+84) le mos le (YINT_M+96) then yms=iym-800-(mos-96);
      if (YINT_M+96) le mos le (YINT_M+108) then yms=iym-900-(mos-108);
      if (YINT_M+108) le mos le (YINT_M+120) then yms=iym-1000-(mos-120);
      if yms<((ys*100)+01) then yms=((ys*100)+01);
      if yms<dliym then yms=dliym;
      if yms>((ys*100)+12) then yms=((ys*100)+12);
    end;

/*weeks missing and continuously receiving - set end date to current interivew date and start date to Dec of start year*/
    else if (-3 le wks le 0 and cwks=1) then do;
      yme=iym; yms=((ys*100)+12); if yms>iym then yms=iym;
    end;

/**weeks missing and not continuously receiving - set end date to December and start date to January of start year**/
    else if (-3 le wks le 0 and cwks=0) then do;
      yms=((ys*100)+01); yme=((ys*100)+12); if yme>iym then yme=iym;
    end;

/**  1.a''  if weeks are known and not currently receiving, then count forward by the number of weeks from January of the
start year. If the count exceeds the interview date then stop counting at the interview date.****/
    else if (wks>0 and cwks=0) then do;
      yms=((ys*100)+01);
      if yms<dliym then yms=dliym;
      if 0 le mos le 12 then yme=(yms+mos);
      if wks>0 and cwks=0 and 13 le mos le 24 then yme=((ys+1)*100)+mos-12;
      if wks>0 and cwks=0 and 25 le mos le 36 then yme=((ys+2)*100)+mos-24;
      if wks>0 and cwks=0 and 37 le mos le 48 then yme=((ys+3)*100)+mos-36;
      if wks>0 and cwks=0 and 49 le mos le 60 then yme=((ys+4)*100)+mos-48;
      if wks>0 and cwks=0 and 61 le mos le 72 then yme=((ys+5)*100)+mos-60;
      if wks>0 and cwks=0 and 73 le mos le 84 then yme=((ys+6)*100)+mos-72;
```

```
      if wks>0 and cwks=0 and 85 le mos le 96 then yme=((ys+7)*100)+mos-84;
      if wks>0 and cwks=0 and 97 le mos le 108 then yme=((ys+8)*100)+mos-96;
      if wks>0 and cwks=0 and 109 le mos le 120 then yme=((ys+9)*100)+mos-108;
      if yme>iym then yme=iym;
    end;
    if yme>0 and cwks=0 then yms=((ys*100)+01);
    if yms<dliym then yms=dliym;
    eflag=2;
  end;
```

/****2. **if start year is unknown but weeks are known then count back from interview date if currently receiving.** If not currently receiving, then count back from interview date to find the most recent year the respondent could have begun receiving and receive for that number of months; then count forward the number of months from January of that year ****/

```
  if -3 le ys le -1 then do;
    if (wks>0 and cwks=1) then do;
      if 0 le mos le YINT_M then yms=iym-mos;
      if YINT_M le mos le (YINT_M+12) then yms=iym-100-(mos-12);
      if YINT_M+12 le mos le (YINT_M+24) then yms=iym-200-(mos-24);
      if (YINT_M+24) le mos le (YINT_M+36) then yms=iym-300-(mos-36);
      if (YINT_M+36) le mos le (YINT_M+48) then yms=iym-400-(mos-48);
      if (YINT_M+48) le mos le (YINT_M+60) then yms=iym-500-(mos-60);
      if (YINT_M+60) le mos le (YINT_M+72) then yms=iym-600-(mos-72);
      if (YINT_M+72) le mos le (YINT_M+84) then yms=iym-700-(mos-84);
      if (YINT_M+84) le mos le (YINT_M+96) then yms=iym-800-(mos-96);
      if (YINT_M+96) le mos le (YINT_M+108) then yms=iym-900-(mos-108);
      if (YINT_M+108) le mos le (YINT_M+120) then yms=iym-1000-(mos-120);
      yme=iym;
    end;
    if (-3 le wks le 0 and cwks=1) then do;  yms=dliym; yme=iym; end;
    if (wks>0 and cwks=0) then do;
      if 0 le mos le YINT_M then do;
        yms=(YINT_Y*100)+01;
        yme=(YINT_Y*100)+01+mos;
      end;
      if (YINT_M+01) le mos le (YINT_M+12) then do;
        yms=((YINT_Y-1)*100)+01;
        if (YINT_M+01) le mos le 12 then yme=((YINT_Y-1)*100)+01+mos;
        if 13 le mos le (YINT_M+12) then yme=(YINT_Y*100)+01+mos;
      end;
      if (YINT_M+13) le mos le (YINT_M+24) then do;
        yms=((YINT_Y-2)*100)+01;
        if (YINT_M+13) le mos le 24 then yme=((YINT_Y-1)*100)+01+mos;
        if 25 le mos le (YINT_M+24) then yme=(YINT_Y*100)+01+mos;
      end;
      if yme>iym then yme=iym;
      end;
    eflag=3;
  end;

  if yme>0 and yms=. then do;
    yms=dliym; if j=1 then eflag=3; else if j>1 then eflag=7;
  end;
```

/****3. **If stop year is known and weeks are known and not currently receiving, but stop month is not known, then count forward from start year.** If the number of months falls short of the stop year, then use January of the end year as the stop date; if the number of months exceeds the stop year, then end the array in the December of the

```
        stop year.  If the stop year is equal to the interview year and the stop month exceeds the interview month, then
        stop at the interview date.  If currently receiving, use interview date as the stop date.  *****/
    if yms>0 and ye>0 and -2 le me le -1 then do;
      if wks>0 then do;
        if 01 le ((yms- round (yms,100))+mos) le 12 then yme=yms+mos;
        if 12 lt ((yms- round (yms,100))+mos) le 24 then yme=yms+100+mos-12;
        if 24 lt ((yms- round (yms,100))+mos) le 36 then yme=yms+200+mos-24;
        if yme>((ye*100)+12) then yme=((ye*100)+12);
        if yme<((ye*100)+01) then yme=((ye*100)+01);
        if yme>iym then yme=iym;
      end;
      if -3 le wks le 0 then yme=((ye*100)+12);
      if yme>iym then yme=iym;
      eflag=4;
    end;

/*** 4.  if stop year is unknown, and weeks are known***/
    if -3 le ye le -1 then do;
      if yms>0 and wks>0 then do;
        if 01 le ((yms- round (yms,100))+mos) le 12 then yme=yms+mos;
        if 12 lt ((yms- round (yms,100))+mos) le 24 then yme=yms+100+mos-12;
        if 24 lt ((yms- round (yms,100))+mos) le 36 then yme=yms+200+mos-24;
        if yme>iym then yme=iym;
      end;

/*** 5.  if stop year is unknown, and weeks are unknown***/
      if yms>0 and -3 le wks le 0 then yme=(round (yms,100))+12;
      eflag=5;
    end;

/*** 6.  if the start and stop years are unknown and the weeks are unknown, use current and last interview date***/
    if -3 le ys le -1 and cwks=0 and -3 le wks le 0 then do;
      yme=iym;
      yms=dliym;
      if j=1 then eflag=6;
      else if j>1 then eflag=7;
    end;

/**     This portion of the program creates a variable that determines who receives afdc in the      **
**      household.  It collapses the answers to 8 categories.  The coding is the following -          **
**              1=respondent only                                                                     **
**              2=spouse/partner only                                                                 **
**              3=child only                                                                          **
**              4=respondent and spouse/partner                                                       **
**              5=respondent and child                                                                **
**              6=spouse/partner and child                                                            **
**              7=respondent and spouse/partner and child                                             **
**              8=other                                                                               **
**      The first 7 categories may include an 'other' person as captured by response categories 4 (other**
**      relatives in household) and 5 (other (specify)) in the original question. If only 'other' person **
**      is listed as receiving, then the 8th answer category is used in the created variable.         **/
    if rcvr=1 and rcvs=0 and rcvc=0  then who=01;        if rcvr=0 and rcvs=1 and rcvc=0  then who=02;
    if rcvr=0 and rcvs=0 and rcvc=1  then who=03;        if rcvr=1 and rcvs=1 and rcvc=0  then who=04;
    if rcvr=1 and rcvs=0 and rcvc=1  then who=05;        if rcvr=0 and rcvs=1 and rcvc=1  then who=06;
    if rcvr=1 and rcvs=1 and rcvc=1  then who=07;
    if -3 le rcvr lt 0 then who=rcvr;
    if rcvr=0 and rcvs=0 and rcvc=0 and (rcvf=1 or rcvo=1) then who=08;
```

```
/**     This portion of the program uses the category reported by the respondent to create an estimated     **
**      amount. The estimated amount is the midpoint rounded down. Note that the 12th category lists     **
**      $1251 as the amount. This amount was chosen since the category is unbounded - the number     **
**      represents one dollar above the lower bound.     **/
    /**food stamps**/
    if -3 le incf1 le -1 and 1 le eincf1 le 10 then incf1=(eincf1*100)-50;
    if -3 le incf1 le -1 and eincf1=11 then incf1=1125;     if -3 le incf1 le -1 and eincf1=12 then incf1=1251;
    if -3 le incf2 le -1 and 1 le eincf2 le 10 then incf2=(eincf2*100)-50;
    if -3 le incf2 le -1 and eincf2=11 then incf2=1125;     if -3 le incf2 le -1 and eincf2=12 then incf2=1251;

    /**afdc**/
    if -3 le inca1 le -1 and 1 le einca1 le 10 then inca1=(einca1*100)-50;
    if -3 le inca1 le -1 and einca1=11 then inca1=1125;     if -3 le inca1 le -1 and einca1=12 then inca1=1251;
    if -3 le inca2 le -1 and 1 le einca2 le 10 then inca2=(einca2*100)-50;
    if -3 le inca2 le -1 and einca2=11 then inca2=1125;     if -3 le inca2 le -1 and einca2=12 then inca2=1251;

    /**other**/
    if -3 le inco1 le -1 and 1 le einco1 le 10 then inco1=(einco1*100)-50;
    if -3 le inco1 le -1 and einco1=11 then inco1=1125;     if -3 le inco1 le -1 and einco1=12 then inco1=1251;
    if -3 le inco2 le -1 and 1 le einco2 le 10 then inco2=(einco2*100)-50;
    if -3 le inco2 le -1 and einco2=11 then inco2=1125;     if -3 le inco2 le -1 and einco2=12 then inco2=1251;

    /**wic**/
    if -3 le incw1 le -1 and 1 le eincw1 le 5 then incw1=(eincw1*20)-10;
    if -3 le incw1 le -1 and eincw1=6 then incw1=101;
    if -3 le incw2 le -1 and 1 le eincw2 le 5 then incw2=(eincw2*20)-10;
    if -3 le incw2 le -1 and eincw2=6 then incw2=101;

    /**unemployment**/
    if -3 le wincu1 le -1 and ewincu1=1 then wincu1=(50/7);
    if -3 le wincu1 le -1 and ewincu1=2 then wincu1=(125/7);
    if -3 le wincu1 le -1 and ewincu1=3 then wincu1=(175/7);
    if -3 le wincu1 le -1 and ewincu1=4 then wincu1=(225/7);
    if -3 le wincu1 le -1 and ewincu1=5 then wincu1=(275/7);
    if -3 le wincu1 le -1 and ewincu1=6 then wincu1=(351/7);
    if -3 le wincu2 le -1 and ewincu2=1 then wincu2=(50/7);
    if -3 le wincu2 le -1 and ewincu2=2 then wincu2=(125/7);
    if -3 le wincu2 le -1 and ewincu2=3 then wincu2=(175/7);
    if -3 le wincu2 le -1 and ewincu2=4 then wincu2=(225/7);
    if -3 le wincu2 le -1 and ewincu2=5 then wincu2=(275/7);
    if -3 le wincu2 le -1 and ewincu2=6 then wincu2=(351/7);

    /**workers comp**/
    if -3 le wincc1 le -1 and 1 le ewincc1 le 10 then wincc1=(((ewincc1*100)-50)/7);
    if -3 le wincc1 le -1 and ewincc1=11 then wincc1=(1125/7);
    if -3 le wincc1 le -1 and ewincc1=12 then wincc1=(1251/7);
    if -3 le wincc2 le -1 and 1 le ewincc2 le 10 then wincc2=(((ewincc2*100)-50)/7);
    if -3 le wincc2 le -1 and ewincc2=11 then wincc2=(1125/7);
    if -3 le wincc2 le -1 and ewincc2=12 then wincc2=(1251/7);

    if inc>-4 then aflag=0;

    csm=(round(yms,100)-198000)*.12+(yms-round(yms,100));
    cem=(round(yme,100)-198000)*.12+(yme-round(yme,100));
    cdli=(round(dliym,100)-198000)*.12+(dliym-round(dliym,100))-1; /***added to zero out people who back report***/
```

```
****to compute the spells of ui;
  if csmu11 ne . and cemu11=. then do; csmu11=.; uiflag=11; end;
  if csmu12 ne . and cemu12=. then do; csmu12=.; uiflag=12; end;
  if csmu13 ne . and cemu13=. then do; csmu13=.; uiflag=13; end;
  if csmu14 ne . and cemu14=. then do; csmu14=.; uiflag=14; end;
  if csmu21 ne . and cemu21=. then do; csmu21=.; uiflag=21; end;
  if csmu22 ne . and cemu22=. then do; csmu22=.; uiflag=22; end;
  if csmu23 ne . and cemu23=. then do; csmu23=.; uiflag=23; end;
  if csmu24 ne . and cemu24=. then do; csmu24=.; uiflag=24; end;

  if 9 le K le 10 then do;
    if (csm gt dlicm) or (csm=dlicm and dli=0) or  (csm le cdli and dli=0) then do;
      if   1 le csm le  12 then nu80=nu80+1;          if  13 le csm le  24 then nu81=nu81+1;
      if  25 le csm le  36 then nu82=nu82+1;          if  37 le csm le  48 then nu83=nu83+1;
      if  49 le csm le  60 then nu84=nu84+1;          if  61 le csm le  72 then nu85=nu85+1;
      if  73 le csm le  84 then nu86=nu86+1;          if  85 le csm le  96 then nu87=nu87+1;
      if  97 le csm le 108 then nu88=nu88+1;          if 109 le csm le 120 then nu89=nu89+1;
      if 121 le csm le 132 then nu90=nu90+1;          if 133 le csm le 144 then nu91=nu91+1;
      if 145 le csm le 156 then nu92=nu92+1;          if 157 le csm le 168 then nu93=nu93+1;
      if 169 le csm le 180 then nu94=nu94+1;          if 181 le csm le 192 then nu95=nu95+1;
      if 193 le csm le 204 then nu96=nu96+1;          if 205 le csm le 216 then nu97=nu97+1;
      if 217 le csm le 228 then nu98=nu98+1;          if 229 le csm le 240 then nu99=nu99+1;
      if 241 le csm le 245 then nu00=nu00+1;
    end;
  end;

****to compute the spells of ui;
  C=0;
  do L=1 to 258;
    C=C+1;
    if 0 le L le doicm then do;
      if csm LE C LE cem then do;
        if 1 le k le 2 then do;
          A=eflag; ahhm=who; aamt=inc; aodd=0;
          if aamt>1000 then do; aodd=aamt; end;
          if A>0 and (aamt=. or ahhm=.) then do; ack=ack+1; end;
          if aamt>0 and (A=0 or ahhm=.) then do; ackm=ackm+1; end;
          if ahhm>0 and (A=0 or aamt=.) then do; ackh=ackh+1; end;
          if aamt=0 then dummya=1;
        end;
        if 3 le k le 4 then do;
          W=eflag; whhm=who; wamt=inc; wodd=0;
          if wamt>1000 then do; wodd=wamt; end;
          if W>0 and (wamt=. or whhm=.) then do; wck=wck+1; end;
          if wamt>0 and (W=0 or whhm=.) then do; wckm=wckm+1; end;
          if whhm>0 and (W=0 or wamt=.) then do; wckh=wckh+1; end;
          if wamt=0 then dummyw=1;
        end;
        if 5 le k le 6 then do;
          F=eflag; fhhm=who; famt=inc; fodd=0;
          if famt>1000 then do; fodd=famt; end;
          if F>0 and (famt=. or fhhm=.) then do; fck=fck+1; end;
          if famt>0 and (F=0 or fhhm=.) then do; fckm=fckm+1; end;
          if fhhm>0 and (F=0 or famt=.) then do; fckh=fckh+1; end;
          if fhhm=. and F>0 and famt ne . then do; fhhm=-3; fckneg=1; end;        /*this fixes a problem in round 3*/
          if famt=0 then dummyf=1;
        end;
```

```
      if 7 le k le 8 then do;
        O=eflag; ohhm=who; oamt=inc; oodd=0;
        if oamt>1000 then do; oodd=oamt; end;
        if O>0 and (oamt=. or ohhm=.) then do; ock=ock+1; end;
        if oamt>0 and (O=0 or ohhm=.) then do; ockm=ockm+1; end;
        if ohhm>0 and (O=0 or oamt=.) then do; ockh=ockh+1; end;
        if oamt=0 then dummyo=1;
      end;
    end;
  end;
end;

C=0;
do L=1 to 258;
  C=C+1;
  if 0 le L le doicm then do;
  if csm LE C LE cem then do;
    do z=1 to 2;
      if 9 le k le 10 then do;
        if (C=csm and inc>0) then do;
          UI=eflag;
          if s_day ne . then uiamt= round ((inc*s_day), 1);
        end;
        else if (C=cem and inc>0) then do;
          UI=eflag;
          if e_day ne . then uiamt= round ((inc*e_day), 1);
        end;
        else if csm<C and C<cem and inc>0 then do;
          UI=eflag;
          if C in (169, 171, 173, 175, 176, 178, 180, 181, 183, 185, 187, 188, 190, 192, 193, 195, 197, 199, 200,
              202, 204, 205, 207, 209, 211, 212, 214, 216, 217, 219, 221, 223, 224, 226, 228, 229, 231, 233, 235,
              236, 238, 240, 241, 243, 245, 247, 248, 250, 252, 253, 255, 257) then do;
           uiamt= round ((inc*31),1);
          end;
          else if C in (172, 174, 177, 179, 184, 186, 189, 191, 196, 198, 201, 203, 208, 210, 213, 215, 220, 222,
              225, 227, 232, 234, 237, 239, 244, 246, 249, 251, 256, 258) then do;
              uiamt=round ((inc*30),1);
          end;
          else if C in (194,242) then do;
              uiamt=round ((inc*29),1);
          end;
          else if C in (170,182,206,218,230,254) then do;
              uiamt= round ((inc*28),1);
          end;
          else if inc<0 then do; UI=eflag; uiamt=inc; end;
        end;
        uiodd=0;
        if uiamt>1000 then do; uiodd=uiamt; end;
        if UI>0 and uiamt=. then do; uick=uick+1; end;
        if uiamt>0 and UI=0 then do; uickm=uickm+1; end;
        if uiamt=0 then dummyui=1;
      end;
    end;
  end;
end;
```

```
C=0;
do L=1 to 258;
  C=C+1;
  if 0 le L le doicm then do;
  if csm LE C LE cem then do;
    do z=3 to 4;
        if 11 le k le 12 then do;
          if (C=csm and inc>0) then do;
            WC=eflag;
            if s_day ne . then wcamt= round ((inc*s_day), 1);
        end;
        else if (C=cem and inc>0) then do;
          WC=eflag;
          if e_day ne . then wcamt= round ((inc*e_day), 1);
        end;
        else if csm<C and C<cem and inc>0 then do;
          WC=eflag;
          if C in (169, 171, 173, 175, 176, 178, 180, 181, 183, 185, 187, 188, 190, 192, 193, 195, 197, 199, 200, 202,
              204, 205, 207, 209, 211, 212, 214, 216, 217, 219, 221, 223, 224, 226, 228, 229, 231, 233, 235, 236,
              238, 240, 241, 243, 245, 247, 248, 250, 252, 253, 255, 257) then do;
                wcamt= round ((inc*31),1);
          end;
          else if C in (172, 174, 177, 179, 184, 186, 189, 191, 196, 198, 201, 203, 208, 210, 213, 215, 220, 222, 225,
              227, 232, 234, 237, 239, 244, 246, 249, 251, 256, 258) then do;
                wcamt=round ((inc*30),1);
          end;
          else if C in (194,242) then do;
                wcamt=round ((inc*29),1);
          end;
          else if C in (170,182,206,218,230,254) then do;
                wcamt= round ((inc*28),1);
          end;
          else if inc<0 then do; WC=eflag; wcamt=inc; end;
        end;
        wcodd=0;
        if wcamt>1000 then do;
          wcodd=wcamt;
        end;
        if WC>0 and wcamt=. then do; wcck=wcck+1; end;
        if wcamt>0 and WC=0 then do; wcckm=wcckm+1; end;
        if wcamt=0 then dummywc=1;
      end;
    end;
  end;
end;

/***(new r4) if start date<current int date and R went through the section previously, zero months post dli***/
C=0;
do L=1 to 258;
  C=C+1;
  if csm le cdli and P010 eq 1 and csm ne . then do;
    if csm le C le cdli then do;
      if 1 le k le 2 then do; A=0; ahhm=.; aamt=.; end;
      if 3 le k le 4 then do; W=0; whhm=.; wamt=.; end;
      if 5 le k le 6 then do; F=0; fhhm=.; famt=.; end;
      if 7 le k le 8 then do; O=0; ohhm=.; oamt=.; end;
```

```
        if 9 le k le 10 then do; UI=0; uiamt=.; end;
        if 11 le k le 12 then do; WC=0; wcamt=.; end;
      end;
    end;
  end;
 end;                                                    /*[e3]*/
end;                                                     /*[e2]*/
                                                         /*2. [e1] end*/
```

/*3. begin - Rs eligible in round 1, round 2 or round 3: used to combine data */

```
array ckm          (i) ckm80-ckm99 ckm00 ckm01;
array ckamt        (i) ckamt80-ckamt99 ckamt00 ckamt01;
array M            (i) M80-M99 M00 M01;
array AMT          (i) AMT80-AMT99 AMT00 AMT01;
array UM           (i) UM80-UM99 UM00 UM01;
array UAMT         (i) UAMT80-UAMT99 UAMT00 UAMT01;
array ckuiamt      (i) ckuiamt80-ckuiamt99 ckuiamt00 ckuiamt01;
array WM           (i) WM80-WM99 WM00 WM01;
array CAMT         (i) CAMT80-CAMT99 CAMT00 CAMT01;
array ckwcamt      (i) ckwcamt80-ckwcamt99 ckwcamt00 ckwcamt01;
array na           (i) na80-na99 na00 na01;
array nn           (i) nn80-nn99 nn00 nn01;
array ua           (i) ua80-ua99 ua00 ua01;
array un           (i) un80-un99 un00 un01;
array wa           (i) wa80-wa99 wa00 wa01;
array wn           (i) wn80-wn99 wn00 wn01;
array oldM         (i) oldM80-oldM99 oldM00 oldM01;
array oldAMT       (i) oldAMT80-oldAMT99 oldAMT00 oldAMT01;
array oldUM        (i) oldUM80-oldUM99 oldUM00 oldUM01;
array oldUAMT      (i) oldUAMT80-oldUAMT99 oldUAMT00 oldUAMT01;
array oldWM        (i) oldWM80-oldWM99 oldWM00 oldWM01;
array oldCAMT      (i) oldCAMT80-oldCAMT99 oldCAMT00 oldCAMT01;
array oldui_spells (i) oldui_spells80-oldui_spells99 oldui_spells00 oldui_spells01;
array oldNU        (i) oldNU80-oldNU99 oldNU00 oldNU01;

do i=1 to 22;
  if gprg_ever>-4 or P1210>-4 or p1210=-5 then do;
    M=0; TTLM=0; AMT=0; UM=0; TTLU=0; UAMT=0; WM=0; TTLW=0; CAMT=0; na=0; nn=0; ua=0; un=0;
        wa=0; wn=0; incprg=0; prgamt=0; nnp=0; nap=0; out=0;
  end;
end;

/*initialize month counter for correction of problematic amounts reported*/
amnths=0; fmnths=0; omnths=0; wmnths=0; uimnths=0; wcmnths=0;

if gprg_ever>-4 or P1210>-4 or p1210=-5 then do;
 do L=1 to 258;
/*calculate total no. months to divide the problematic amts over*/
   if A>0 and aamt>1000 and aodd>0 then do; amnths=amnths+1; end;
   if F>0 and famt>1000 and fodd>0 then do; fmnths=fmnths+1; end;
   if O>0 and oamt>1000 and oodd>0 then do; omnths=omnths+1; end;
   if W>0 and wamt>1000 and wodd>0 then do; wmnths=wmnths+1; end;
   if UI>0 and uiamt>1000 and uiodd>0 then do; uimnths=uimnths+1; end;
   if WC>0 and wcamt>1000 and wcodd>0 then do;  wcmnths=wcmnths+1; end;
  end;
```

```
end;

if gprg_ever>-4 or P1210>-4 or p1210=-5 then do;
  do L=1 to 258;
/*divide the amount by the number of months*/
    if aamt>1000 and amnths>0 and A>0 then do; aamt=round(aamt/amnths, 1); end;
    if famt>1000 and fmnths>0 and F>0 then do; famt=round(famt/fmnths, 1); end;
    if oamt>1000 and omnths>0 and O>0 then do; oamt=round(oamt/omnths, 1); end;
    if wamt>1000 and wmnths>0 and W>0 then do; wamt=round(wamt/wmnths, 1); end;
    if uiamt>1000 and uimnths>0 and UI>0 then do; uiamt=round(uiamt/uimnths, 1); end;
    if wcamt>1000 and wcmnths>0 and WC>0 then do; wcamt=round(wcamt/wcmnths, 1); end;

/*1980*/
    if 1 le L le 12 then do;
      if M80 ge 0 and A>0 or W>0 or F>0 or O>0 then do; M80=M80+1; end;
      if A>1 then M80=-3;
      if W>1 then M80=-3;
      if F>1 then M80=-3;
      if O>1 then M80=-3;
      array allamt80 aamt wamt famt oamt;
      do over allamt80;
        if -3 le allamt80 le -1 then do; nn80=nn80+1; amt80=allamt80; end;
        if allamt80 ge 0 then do;
          na80=na80+1;
          if na80=1 and nn80=0 then amt80=allamt80;
          if na80>1 and nn80=0 then amt80=amt80+allamt80;
          if M80=-3 then amt80=-3;
        end;
      end;

      if UM80 ge 0 and UI>0 then do; UM80=UM80+1; end;
      if UI>1 then UM80=-3;
      if -3 le uiamt le -1 then do; un80=un80+1; UAMT80=uiamt; end;
      if uiamt ge 0 then do;
        ua80=ua80+1;
        if ua80=1 and un80=0 then UAMT80=uiamt;
        if ua80>1 and un80=0 then UAMT80=UAMT80+uiamt;
        if UM80=-3 then UAMT80=-3;
      end;

      if WM80 ge 0 and WC>0 then do; WM80=WM80+1; end;
      if WC>1 then WM80=-3;
      if -3 le wcamt le -1 then do; wn80=wn80+1; CAMT80=wcamt; end;
      if wcamt ge 0 then do;
        wa80=wa80+1;
        if wa80=1 and wn80=0 then CAMT80=wcamt;
        if wa80>1 and wn80=0 then CAMT80=CAMT80+wcamt;
        if WM80=-3 then CAMT80=-3;
      end;
    end;
```

/*At this point the code repeats for each year 1981-2001. This code is not included here due to space considerations. Contact NLS User Services for information. The month numbers for each year are as follows:

1981	13-24	1988	97-108	1995	181-192
1982	25-36	1989	109-120	1996	193-204
1983	37-48	1990	121-132	1997	205-216
1984	49-60	1991	133-144	1998	217-228
1985	61-72	1992	145-156	1999	229-240
1986	73-84	1993	157-168	2000	241-252
1987	85-96	1994	169-180	2001	253-264

```
*total months - part 2;
   if 1 le L le 258 then do;
     if TTLM ge 0 and A>0 or W>0 or F>0 or O>0 then do; TTLM=TTLM+1; end;
     if A>1 then TTLM=-3; if W>1 then TTLM=-3; if F>1 then TTLM=-3; if O>1 then TTLM=-3;
     if TTLU ge 0 and UI>0 then do; TTLU=TTLU+1; end;
     if UI>1 then TTLU=-3;
     if TTLW ge 0 and WC>0 then do; TTLW=TTLW+1; end;
     if WC>1 then TTLW=-3;
   end;


**for income program amounts;
     if 229 le L le 240 then do;
       if W>0 or F>0 then do; out=1; end;
       if incprg ge 0 and (A>0 or O>0 or UI>0 or WC>0) then do; incprg=incprg+1; end;
       if A>1 or O>1 or UI>1 or WC>1 then incprg=-3;
       array prgamts aamt oamt uiamt wcamt;
       do over prgamts;
         if -3 le prgamts le -1 then do; nnp=nnp+1; prgamt=prgamts; end;
         if prgamts ge 0 then do;
           nap=nap+1;
           if nap=1 and nnp=0 then prgamt=prgamts;
           if nap>1 and nnp=0 then prgamt=prgamt+prgamts;
           if incprg=-3 then prgamt=-3;
         end;
       end;
     end;
  end;
end;                                                        /*3. end*/

do i=1 to 22;
   if gprg_ever=. then gprg_ever=0;
   oldM=gtotal; oldAMT=gatotal; if gtotal=-4 then gtotal=0; if gatotal=-4 then gatotal=0;
   if M ge 0 and gtotal ge 0 then do; gtotal=M+gtotal; end;
   if M<0 or gtotal<0 then do; gtotal=-3; end;
   if amt ge 0 and gatotal ge 0 then do; gatotal=AMT+gatotal; end;
   if AMT<0 or gatotal<0 or (gtotal<0 and gtotal ne -4) then do; gatotal=-3; end;
   if AMT=0 and oldAMT=-4 and gtotal=0 then do; gatotal=-4; end;
end;

*total months, part 2;
if TTLM ge 0 and gprg_ever ge 0 then do; TTLM=TTLM+gprg_ever; end;
if TTLM<0 or gprg_ever<0 then do; TTLM=-3; end;
if TTLU ge 0 and ui_ever ge 0 then do; TTLU=TTLU+ui_ever; end;
if TTLU<0 or ui_ever<0 then do; TTLU=-3; end;
if TTLW ge 0 and wc_ever ge 0 then do; TTLW=TTLW+wc_ever; end;
if TTLW<0 or wc_ever<0 then do; TTLW=-3; end;
```

```
*calculating the unemployment spells;
do i=1 to 22;
   oldui_spells=ui_spells;
   oldnu=nu;
   if ui_spells=-4 then ui_spells=0;
   if NU ge 0 and ui_spells ge 0 then do; NU=NU+ui_spells; end;
   if (ui_spells ge -3 and ui_spells le -1) then do; NU=-3; end;
   if NU=0 and oldui_spells=-4 then do; NU=-4; end;
end;

do i=1 to 22;
   oldUM=uitotal;                    oldUAMT=uiatotal;
   if uitotal=-4 then uitotal=0;     if uiatotal=-4 then uiatotal=0;
   if UM ge 0 and uitotal ge 0 then do; uitotal=UM+uitotal; end;
   if UM<0 or uitotal<0 then do; uitotal=-3; end;
   if UAMT ge 0 and uiatotal ge 0 then do; uiatotal=UAMT+uiatotal; end;
   if UAMT<0 or uiatotal<0 or uitotal<0 then do; uiatotal=-3; end;
   if UAMT=0 and oldUAMT=-4 and uitotal=0 then do; uiatotal=-4; end;
end;

do i=1 to 22;
   oldWM=wctotal;                    oldCAMT=wcatotal;
   if wctotal=-4 then wctotal=0;     if wcatotal=-4 then wcatotal=0;
   if WM ge 0 and wctotal ge 0 then do; wctotal=WM+wctotal; end;
   if WM<0 or wctotal<0 then do; wctotal=-3; end;
   if CAMT ge 0 and wcatotal ge 0 then do; wcatotal=CAMT+wcatotal; end;
   if CAMT<0 or wcatotal<0 or wctotal<0 then do; wcatotal=-3; end;
   if CAMT=0 and oldCAMT=-4 and wctotal=0 then do; wcatotal=-4; end;
end;

do i=1 to 22;
   if p1210=-4 then do; gtotal=-4; gatotal=-4; TTLM=-4; uitotal=-4; NU=-4; uiatotal=-4; TTLU=-4; wctotal=-4;
         wcatotal=-4; TTLW=-4; incprg=-4; prgamt=-4; end;
   if p1210=-5 then do; gtotal=-5; gatotal=-5; TTLM=-5; uitotal=-5; NU=-5; uiatotal=-5; TTLU=-5; wctotal=-5;
         wcatotal=-5; TTLW=-5; incprg=-5; prgamt=-5; end;
end;

/*To change all invalid skips to -3*/
array vars gtotal80-gtotal99 gtotal00 gtotal01 gatotal80-gatotal99 gatotal00 gatotal01
uitotal80-uitotal99 uitotal00 uitotal01 uiatotal80-uiatotal99 uiatotal00 uiatotal01
NU80-NU99 NU00 NU01 TTLM TTLU wcatotal80-wcatotal99 wcatotal00 wcatotal01;

do over vars;
   if vars=-2 then vars=-3;
   if vars=-1 then vars=-3;
end;

endsas;
```

NLSY97 Appendix 6:

Event History Creation and Documentation

The NLSY97 survey records significant life-course transitions experienced by young people, such as education, employment, program participation, and marital history, in a longitudinal format. The event history arrays document these events in a chronological format that records the significant transitions in a meaningful manner while maintaining data quality. Using these arrays, researchers can extract the status of a respondent at a point in time or over time. Event history arrays are generated for four distinct areas: employment, marital/cohabitation status, program participation, and schooling. This section presents information on each type of event history array; for details on the chronological format of the arrays and the naming conventions used to identify the variables, users should refer to Appendix 7.

EMPLOYMENT EVENT HISTORY ARRAYS

Three employment arrays provide information on the respondent's civilian employment on a weekly basis. These arrays include information about employee jobs only; freelance jobs are not included in the arrays. All employment arrays provide information starting in the month when the respondent turned 14 and ending in the week that he or she was last interviewed.

1. EMP_STATUS

This main array presents the civilian employment status of a respondent in a particular week. The codes and their explanations follow:

Code	Definition
Status=0: No information reported to account for week	Week cannot be assigned due to missing job start and stop dates.
Status=1: Not associated with an employer, not actively searching for an employee job	Refers to weeks during a between-jobs gap in which the respondent is not actively searching and reports working at a freelance job. Since the actual weeks working at a freelance job cannot be determined, all weeks in which the respondent is not actively searching are coded in this manner. Similar information was not collected for the within-job gaps during round 1.
Status=2: Not working (unemployment vs. out of labor force cannot be determined)	Assigned when the respondent is not asked follow-up questions about his or her search activity during a within-job gap or a between-jobs gap.
Status=3: Associated with an employer, periods not working for employer are missing	Used when a respondent reports an indeterminate start or stop date for a within-job gap.
Status=4: Unemployed	Indicates that the respondent reports actively searching for work during a within-job gap or a between-jobs gap. When the number of weeks unemployed do not account for the entire gap period, weeks unemployed are assumed to occur in the middle of that period.
Status=5: Out of the labor force	Assigned during a between-jobs gap or a within-job gap when the respondent is either not actively searching for work or on layoff from a job.
Status=6: Active military service	Indicates that the respondent is tied to the military.
Status=9701 to 9809: Employer on roster	Refers to the employer number on the employer roster (YEMP_UID.01 to YEMP_UID.09). Presence of an employer number indicates that the respondent was working during a given week. Civilian work takes precedence over other activities, such as job search. Respondents who report working at an employer job for one day in a given week are listed as having worked at that job for the entire week, regardless of other activities.

2. EMP_DUAL_JOB#
If a respondent holds more than one civilian employee job during a week, the second employee job is presented in a dual job array. These arrays contain only the job number of the overlapping job; labor force status information is only included in the main array. For example, if a respondent held two employee jobs (e.g., the first and third jobs listed on the employer roster) in one week, the employer number for the first job would be recorded in the EMP_STATUS array and the employer number for the third job would be recorded in the EMP_DUAL_2 array. If a respondent held three jobs (e.g., jobs #01, #04, and #05 on the roster) in one week, the first job would be recorded in the EMP_STATUS array, the employer ID for job #04 would be recorded in the EMP_DUAL_2 array, and the employer ID for job #05 would be recorded in the EMP_DUAL_3 array.

3. EMP_HOURS
This final array calculates the total number of hours worked by a respondent at any civilian employee job during each week. Hours per week worked at each job are assumed constant except during a reported gap, when the hours for that job are assumed to be zero. Each week is assigned a code of '-3 (invalid skip)' when any of the jobs has an indeterminate gap date.

Other Information
Continuous week crosswalk. A secondary set of variables translates the reported beginning and ending dates (day, month, and year) of employee jobs and the gaps within those jobs to the week and year naming scheme (e.g., EMP_GAP_START_YEAR.01.01 and EMP_GAP_END_YEAR.01.01 provide the start and end dates of the respondent's first gap at the first job in the continuous week and year format). More information about the week and year naming scheme is provided in appendix 7 in this document.

Linking to survey data using unique ID codes. The created event history variables can be used in conjunction with the main file information about the respondent's employment. The 9701–9707, 9801–9809, 199901–199909, and 200001–200009 codes that appear in the status and dual job arrays are unique employer ID numbers that are also used in the main file data. In the main data, these codes are listed in the YEMP_UID.xx variables (e.g., R24761.). Using these unique ID codes, researchers can identify the comparable job information (e.g., complete start and stop dates, fringe benefits, job satisfaction, industry and occupation, etc.) from the main file. Employers first reported in the round 1 interview have codes beginning with "97"; new employers in round 2 were assigned codes beginning with "98." Beginning in round 3 the ID codes were assigned using a 4-digit year, so each round 3 code begins with "1999" and round 4 codes begin with "2000." This system permits users to link employers across survey rounds and across data files and to identify the round in which an employer was first reported.

> ***User Notes:*** The collection of freelance and self-employment information changed in the round 4 interview, as described in the introduction to appendix 2. A small number of round 4 self-employed jobs may have a unique ID of 199999. The assignment of unique ID codes is described in detail in appendix 8 in this codebook supplement.

Denial of previously reported employers. Respondents sometimes deny that they ever worked for an employer reported in a previous round. If this situation occurs, the data for that employer remain in the event history arrays, but a flag variable (EMP_DENY) indicates that the employer was denied in a subsequent round. For example, assume that a respondent reported working for employer number 9802 from January 1, 1998 through the round 2 interview date. In round 3, however, the respondent stated that he or she never worked for that employer. The weekly STATUS variables for January 1, 1998, through the round 2 interview date will continue to report the respondent's status as working for employer 9802, but the EMP_DENY variable will also have a value of 9802, indicating that the respondent denied working for that employer during the round 3 interview.

Missing and imputed values. Occasionally, respondents cannot provide information about the start and end dates of employment periods or gaps in employment. Because dates of employment are often used in subsequent questions in the jobs section, default values are substituted for these missing values so that the interview program can continue. The missing values are then reinserted in the public use data file so that researchers will know the true value. However, to follow the flow of an interview, users may need to understand what values were substituted so that the correct question path can be followed. Similarly, in the creation of the event history arrays, some missing values are imputed. Imputation rules and the effect of each on the event history arrays are as follows:

Type of missing data	Imputed value in interview and event history data	Effect on event history variables
Missing job start or stop day	Start day = 1 Stop day = 28	If there is a valid month and year, the imputed days are used in the creation of status variables as if they were valid data. For example, a respondent with an imputed start day of "1" for employer 9701 will be listed as working for employer 9701 for the first week (and each subsequent week) of the reported month in the STATUS array.
Missing job start or stop month	Start month = 1 Stop month = 12	In the STATUS array, weeks in imputed months are assigned a status of "0"—no information. Each month from the beginning of the job to the next known date or from the last known date to the end of the job is assigned a 0. For example, assume a respondent reports starting a job in an unknown month of 1997 but then reports a within-job gap starting on 6/1/97. All weeks in the months of January–May will be assigned a value of 0.
Missing job start or stop year	Start year = year of last interview Stop year = year of current interview	In the STATUS array, weeks in imputed years are assigned a status of 0. For example, in round 2 a respondent who reported a new job with an unknown start year would be assigned an imputed value equal to the year of the round 1 interview (usually 1997). In the STATUS variables, each week from the round 1 interview date to the first known employment date (or the current interview date) would have a value of 0.
Missing gap start or stop day	Start day = 1 Stop day = 28	If there is a valid month and year, the imputed days are used in the creation of status variables as if they were valid data. For example, a respondent with an imputed start day of "1" for a within-job gap will be assigned a value of 2, 4, or 5—depending on information about layoff and job search—for the first week (and each subsequent week) of the reported month in the STATUS array.
Missing gap start or stop month; missing gap start or stop year	Start month = job start date (or the date of a previous known gap) Stop month = job stop month (or the date of a later known gap)	Each week in the imputed period is assigned a value of 3 in the STATUS array, meaning associated with an employer but with missing gap information. For example, a respondent with a job start date of 4/12/97 and a gap with an unknown start month and year would have an imputed gap start date of 4/12/97. Each week from that date to the next known employment date would have a status of 3.

Respondents may have more than one job in a given week due to the imputation of dates as described above. If the month or year was imputed for one job, resulting in the assignment of zeros to a given set of weeks, but another job with known dates falls in some of those weeks, the zeros will be dropped and replaced by information about the known job. However, the respondent will **not** be listed as having a dual job in those weeks. The imputed employer will not be listed in any array if the zeros are dropped because another job provides valid information.

Backreporter variables. Some respondents report during the current interview a new job with a start date prior to the date of the last interview that was not reported during that interview. If these jobs had been reported at the previous interview, the weeks and hours worked would have been represented in the arrays at that time. When they are instead reported in the current interview, the event history arrays created at the previous interview date are not changed to include information about these new jobs. Three "backreporter" variables alert users to changes that would have resulted if the jobs had been correctly reported during the previous interview.

The first variable, EMP_BK_WKS, tells how many weeks before the previous interview date the job started. The second and third variables show how the status and hours arrays would have been affected had the job beginning before the date of last interview been reported at the prior interview and included in the original array construction. One variable, EMP_BK_STATUS, indicates the number of weeks from the job's start date to the date of last interview for which a nonworking status would have changed to an employer ID had the job been reported during the previous interview round. The other variable, EMP_BK_HOURS, informs users about the additional number of hours per week worked on this job for the weeks from the job's start date to the date of the previous interview.

For example, assume a respondent named Mary was interviewed on January 15, 1999 (round 3), and January 15, 2000 (round 4). In round 3, Mary reported no employers. In round 4, she reported working 30 hours a week on a job that began on January 1, 1999. Since the job began 2 weeks before the round 3 interview, EMP_BK_WKS would have a value of 2. EMP_BK_STATUS would also have a value of 2, indicating that 2 weeks in the round 3 arrays would have changed from nonworking to working status. EMP_BK_HOURS would have a value of 30, indicating that 30 additional hours would have been worked in each of those weeks.

Similarly, assume a respondent named John was interviewed on the same dates as Mary in rounds 3 and 4. In round 3, John reported a job that he had worked at for 10 hours per week since the round 2 interview. In round 4, he reported a second, 20 hours-per-week job that began on January 1, 1999, 2 weeks before his round 3 interview. Like Mary, John would have a value of 2 for the EMP_BK_WKS variable. However, the weeks between January 1 and January 15, 1999, would already indicate that John was working (at the original employer). Therefore, EMP_BK_STATUS would have a value of 0, because no weeks would have changed from nonworking to working status if John had reported the new job in round 3. EMP_BK_HOURS would have a value of 20, indicating the number of hours per week that John worked at the new job. In John's case, the hours worked array variables created in round 3 would have a value of 10, reflecting the job he reported in round 3. Researchers can add the value of EMP_BK_HOURS to the value in the original round 3 arrays for the 2 weeks before January 15, 1999, to determine that John worked 30 hours per week in those weeks.

MARITAL STATUS EVENT HISTORY ARRAYS

The NLSY97 marital and cohabitation arrays record changes in the respondent's marital status and cohabitation changes on a monthly basis. The marital/cohabitation history program converts dates reported in the marriage section (beginning and ending dates of cohabitations, marriages, separations, divorces, and widowhoods) to an actual month number, using January 1980 as month #1. Used jointly, these arrays allow the researcher to obtain a detailed history of the respondent's partners and changes in his/her marital and cohabitation status on a monthly basis. All marital/cohabitation arrays provide information beginning in the month that the respondent turned 14 (although respondents do not answer marriage and cohabitation questions until they reach age 16) and ending in the month that he or she was last interviewed. Additionally, the beginning dates of the youth's first marriage and first cohabitation are provided in two variables: CV_FIRST_MARRY_MONTH and CV_FIRST_COHAB_MONTH. Note that the continuous month variables reporting the births of the respondent's children (e.g., CV_CHILD_BIRTH_MONTH.01) permit easy comparison of the timing of marital and fertility events. These variables are described in Appendix 3.

Three types of arrays record transitions between living without a partner of the opposite sex to cohabiting or to marriage.

1. MAR_STATUS
The main array presents the status (e.g., never married/not cohabiting, cohabiting, married, divorced) of a respondent during a particular month. Marital status takes precedence over cohabiting; for example, if a respondent is divorced and living with another partner, the status listed in this array will be 'divorced.' Respondents who are married but not living with their spouse are coded as married. If a respondent reports an annulment, the previously reported dates of marriage are maintained and the marital status code after the annulment is 'divorced.'

2. MAR_COHABITATION
This second array details the partner that the respondent is living with in a particular month. For example, if the respondent is cohabiting, the variable for each month identifies whether the respondent lives with partner 1, partner 2, spouse 1, spouse 2, etc. Users should note that "1" and "2" in this case refer to the respondent's partners/spouses in chronological order. The numbers do not necessarily refer to the same person as the loop numbers in the spouse/partner questions asked directly of the respondent during the survey. Users can distinguish between partners and spouses because partner IDs begin with "1" (e.g., 101, 102) and spouse IDs begin with "2" (e.g., 201, 202).

3. MAR_PARTNER_LINK
The third array links the cohabiting partner or spouse to the partner order in the main survey questions. This array allows the researcher to identify characteristics of the respondent's partner and to link them with spells of marriage or cohabitation. For example, a researcher might look at the MAR_COHABITATION variable for the 9th month of 1998 and determine that a respondent was living with his second partner ever in that month because the variable's value is 102. If the researcher checks the value of MAR_PARTNER_LINK for the same month and year, the respondent might have a value of 9801, indicating that the partner in the event history arrays that month is the first partner reported in the round 2 survey. The researcher can then examine the round 2 variables with "Spouse/Partner 01" in the title to determine that person's characteristics, such as race, ethnicity, age, religion, and so on. However, if there is a significant gap between relationship spells—for example, if the respondent was married and then divorced a spouse before round 1 and then began cohabiting with the same person in round 3—the survey would not necessarily identify this as the same person.

Other information
MAR_DUAL. Rounds 1 and 2 contain a fourth monthly array. If there is an overlap of partners (e.g., partner 1 leaves at the beginning of the month and partner 2 moves in at the end of the month), this array records the presence of the new partner. The format of these variables is the same as the MAR_COHABITATION variables. Beginning in round 3, this array was changed. Because this is a relatively rare event, there is only one variable per round that indicates whether there was any month with an overlap period between the current interview and the previous interview.

Denial of previous data. Occasionally, respondents report that the marital status information from a previous round is not true. As is the case with employment, the information in arrays based on that rounds' information is maintained, but the respondent is assigned a value in a flag variable (MAR_DENY) indicating later denial of the information. This flag variable has several different possible values, depending on the type of information denied. For example, assume a respondent reported cohabiting at the round 2 interview date but denied that the cohabitation had occurred in round 3. The status variables for each month from the beginning of the cohabitation to the round 2 interview date would continue to reflect the cohabitation, but the MAR_DENY variable would have a value of 3, indicating that the cohabitation was later denied.

PROGRAM PARTICIPATION EVENT HISTORY ARRAYS

Program participation arrays are constructed individually for five programs—Worker's Compensation, Unemployment Insurance, AFDC, Food Stamps, and WIC. The AFDC array includes all federal and state programs created under Temporary Assistance to Needy Families (TANF) or any government program for needy families that replaces AFDC. All other programs (e.g., LIHEAP, SSI, other) are combined into a sixth array entitled 'Other.' For each program type, except Worker's and Unemployment Compensation, three arrays are created. All program participation arrays provide information starting in the month that the respondent turned 14 and ending in the month that he or she was last interviewed.

A secondary set of variables translates the reported beginning and ending dates (month and year) of a spell within the program into the continuous month scheme (e.g., AFDC_START_MONTH and AFDC_STOP_MONTH). More information about the continuous month scheme is provided in Appendix 7 in this document.

1. STATUS

The main array, (e.g., AFDC_STATUS), presents the status—receiving or not—of a respondent during each month. When asked for the start or stop date of a spell, the respondent could answer 'don't know' or 'refuse' to any component. In this case, the respondent was then asked how many weeks the spell lasted. The number of reported weeks was then divided by 4.3 to determine the equivalent number of months. If a fraction of a month was reported, then the entire month was counted as a month receiving benefits. Using a combination of start date, stop date, and week information, each spell was defined and a value of '1' inserted into the status array to indicate months of receipt. The months that a respondent did not receive that benefit, but was eligible to receive it, have a value of '0.' An edit variable (e.g., AFDC_EDIT_DATE) flags respondent-reported and imputed dates. The process by which imputed dates and the corresponding edit flag were assigned is described below:

Flag	Definition
Edit Flag=1: Respondent reported participation dates	Respondent reported a complete start and stop date and is not currently receiving. If the respondent reports still receiving at the time of the interview, the interview date is assigned as the temporary stop date. In the next survey round, the respondent will be asked if he or she is still receiving; if not, a permanent stop date equivalent to the previous round's interview date will be assigned. If the respondent reports receiving, participation will continue in filling the array.
Edit Flag=2: Start month imputed	**Total weeks known:** If the respondent reports not currently receiving, then set the month equal to January and count forward by the number of weeks to imply a stop date. If currently receiving, then count back by the number of weeks from the interview date to impute a start month. If the month indicated by the count falls short of the start year, the start month is December of the start year. If the month occurs in the year before the reported start year, then the start month is January of the start year.
	Total weeks unknown: If the respondent reports not currently receiving, then the start month is set to January. Use December as the stop month and the start year as the stop year. If the respondent reports currently receiving, use December as the start month.
Edit Flag=3: Start month and year imputed	**Total weeks known:** Count back by the number of weeks from the interview date if currently receiving. If not currently receiving, then count back from interview date to find the most recent year the respondent could have begun receiving and call the start date January of that year; then count forward the number of weeks from that date to imply a stop date.
	Total weeks unknown: If currently receiving, begin the spell at the respondent's 14th birthday.

Flag	Definition
Edit Flag=4: Stop month imputed	**Total weeks known:** If not currently receiving, then count forward from start date. If the month indicated falls short of the stop year, then use January of the stop year as the stop month; if the number of months exceeds the stop year, then set the stop month to December of the stop year. If the stop year is equal to the interview year and the stop month exceeds the interview month, then stop at the interview date.
	Total weeks unknown: If not currently receiving, then use December of stop year for the stop month.
Edit Flag=5: Stop month and year imputed	**Total weeks known:** If not still receiving, count forward from the start date.
	Total weeks unknown: If not currently receiving, then use December of the start year as the stop month and the start year as the stop year.
Edit Flag=6: Start and stop dates imputed	**Total weeks unknown:** The imputed dates are based on the previous interview's date (start date) to the current interview date (stop date); in round 1, the last interview date is the respondent's 14th birth month and year.
Edit Flag=7: Start and stop dates complete but gap information missing	In these cases, the respondent began a spell of receipt in round 1 and ended it in round 2. The start and stop dates are accurate, but no information was collected about gaps in receipt.

2. AMOUNT RECEIVED

If a respondent reports receiving in a particular month, a second array presents the amount received in each month (e.g., AFDC_AMT). The dollar values asked about during the interview were meant to be monthly values. However, some responses were higher than the federal or state limits on the amount received from a particular benefit. A likely reason is that the respondent mistakenly reported a total value rather than a monthly value. Values determined to be too high were divided by the number of months the respondent reported receiving the benefit. These values were used in the AMT arrays instead. A second set of edit variables (e.g., AFDC_EDIT_AMT) flags these values for a particular spell. In round 1, the edit values were set for the latest year available. For AFDC, the highest benefit offered was $1229; reported amounts falling above this maximum were edited. The maximum amount accepted for food stamps was $936; the edit flag indicates reported amounts that fall above this maximum.

3. HOUSEHOLD MEMBERS RECEIVING

If a respondent reports receiving in a particular month, the persons in the household who benefit from the program in each month (e.g., respondent only, child only, respondent and child) are recorded in a third array (e.g., AFDC_HH). This program condenses the set of answers from the question in the survey that collects this information; for example, see YPRG-18300.01_001 to YPRG-18300.01_005 for AFDC. Users should note that this array is not present for Worker's Compensation and Unemployment Insurance because these programs are collected for the respondent only.

Other information

A few respondents report receiving assistance but then deny that receipt in a later interview. These situations are treated in the same way as in the marriage arrays, as described above. The denial flag variables for the program participation arrays incorporate the name of the program (e.g., AFDC_DENY).

Researchers should be aware of an important source of variability in the Worker's Compensation data. These data suggest that some respondents report the dates the payment was actually received and some report the period of time to which the payment applied. For example, if a respondent was out of work for six months but received a lump sum payment a year later, he or she might report either the date the lump sum was paid or the dates he or she was unable to work.

SCHOOLING EVENT HISTORY ARRAYS

These education arrays are somewhat different than the other event history arrays. Information on a respondent's education is reported in both yearly and monthly variables. This approach is used to combine information from the youth questionnaire, which collects more detailed data, and from the round 1 parent questionnaire, which presented information only for each year. Users should be aware that, because questions were not identical in the round 1 parent questionnaire and the round 2 youth questionnaire, the transition between the two data sources was not seamless and some information for the yearly variables had to be imputed. If they feel that a given value is questionable, researchers may wish to compare created yearly variables to the raw data and to the monthly schooling arrays described below.

Yearly Schooling Variables: A set of schooling variables provides information for each year beginning in 1980, the year when the first information is available in the survey, through the most recent interview year. In general, these variables refer to the school year rather than the calendar year. That is, 1991 in a variable title or in the data for a variable generally indicates the school year starting in fall 1991 and ending in spring 1992.

1. SCH_YEAR_to_GRADE

This array presents the grade the respondent attended during the school year. The last four digits of the question name indicate the school year. For example, SCH_YEAR_to_GRADE.1990 refers to the grade attended by the respondent during the school year that starts in fall 1990 and ends in spring 1991.

2. SCH_GRADE_to_YEAR

This array refers to the year the respondent attended a certain grade. For example, if the respondent attended second grade in 1992–93, then SCH_GRADE_to_YEAR.2 would have the value 1992.

3. SCH_CHANGES

This array counts the number of times the respondent changed the school attended during the school year. For example, SCH_CHANGES.1990 shows how many different schools the respondent attended during the school year that started in fall 1990 and ended in spring 1991.

4. SCH_MNTHS_MISSED

This array presents the number of months during the school year that the respondent did not attend school. For example, if SCH_MNTHS_MISSED.1990 has a value of 3 for a respondent, then that respondent had a gap in attendance of three months during the school year that started in the fall of 1990 and ended in the spring of 1991. A gap is defined as missing school for one or more months (not including summer vacation); gaps do not have to be consecutive.

5. SCH_SUMMER_SCHOOL

This array refers to extra school classes during an educational break in a given school year, such as summer school. For example, SCH_SUMMER_SCHOOL.1990 shows whether the respondent attended school during a break in the 1990–91 school year.

6. SCH_GRADE_PROGRESS

This array has positive values if there are any special events that occurred during the school grade. For example, a positive value in SCH_GRADE_PROGRESS.2 indicates that the respondent was skipped or demoted during second grade. Researchers should note that parents might have been confused as to how to answer the skip grade questions asked during the interview. For example, there are parents who say their child skipped from 5th to 6th grade, while others say from 4th to 6th grades. Both of these cases are probably stating that the child missed most or all of the 5th grade. To resolve this ambiguity, the code states that if a child is skipped consecutive years then the first year (i.e. 5th grade) was missed. If a parent reports non-consecutive years (i.e. 4th to 6th) then the program assumes the year(s) in the middle are the ones not attended.

7. SCH_YEAR_PROGRESS
This array refers to any special events that occurred during the school year. The question name's last four digits indicate the school year this variable refers to. For example, SCH_YEAR_PROGRESS.1990 shows special events that occurred during the school year that starts in fall 1990 and ends in spring 1991. The special events, such as grades skipped or demoted to, are defined in the same way as in the previous array.

8. SCH_SUSPENSIONS
This array counts the number of days during the school year the respondent was suspended from school. For example, if SCH_SUSPENSIONS.1990 has a value of 3 then the respondent was suspended from school 3 days during the school year that started in fall 1990 and ended in spring 1991.

Monthly Schooling Variables. In rounds 2–4, three types of monthly arrays and one flag variable (SCH_DUAL) were created. Each array captures information for each month from the respondent's interview date in round 1 to the most recent interview date.

1. SCH_STATUS
This array reports the respondent's enrollment status during each month from the round 1 interview date through the current interview date. Coding categories include unknown, not enrolled, in grades K to 12, in college, on vacation, expelled, and other.

2. SCH_TERM
These variables report the respondent's school type and grade for each month in the time period. The first two digits represent the type of school (public = 10, private = 20, and religious = 30). The last two digits provide the respondent's grade in school (1–12) or year in college (1–8). To determine whether the school is a K–12 school or a college, researchers are advised to combine this variable with the SCH_STATUS variable described above.

3. SCH_ID
This variable permits users to link array information to the school roster in the main data file and access other information about the school. The variable uses the same ID codes as the identification variable on the school roster in the main data set (for example, NEWSCHOOL_PUBID.01).

4. SCH_DUAL
A small number of NLSY97 respondents went to two different schools in the same month. Because only the first school can be reported in the other arrays, this variable flags these special cases. There is only one variable for the period between each round (that is, round 1 to round 2, round 2 to round 3, etc.); the exact month when the overlap occurred is not indicated, and overlap may have occurred in more than one month.

Flag	Definition
Flag=1	More than one K–12 school attended in the same month
Flag=2	More than one college attended in the same month
Flag=3	More than one K–12 school attended in the same month and more than one college attended in the same month
Flag=4	K–12 school and college attended in the same month
Flag=5	More than one K–12 school attended in the same month and K–12 school and college attended in the same month
Flag=6	More than one college attended in the same month and K–12 school and college attended in the same month
Flag=7	More than one K–12 school attended in the same month and more than one college attended in the same month and K–12 school and college attended in the same month

NLSY97 Appendix 7:

Continuous Month Scheme and Crosswalk

As described in appendix 6, the various event history arrays present month-by-month or week-by-week information on the respondent's status with respect to various activities. Information about program participation, marriage, and schooling is provided on a monthly basis using a continuous month timeline. Employment histories are presented on a weekly basis using a continuous week timeline. This appendix explains the continuous month/week system and provides the actual dates corresponding to the numbered weeks and months.

Continuous Month Timeline

The non-employment event history arrays are presented using two different timelines. The first, a continuous month approach, labels January 1980 as month 1, February 1980 as month 2, and so on. Thus, a respondent born in month 4 might start receiving public assistance in month 193 and leave the program in month 198. Key events occurring during the life of each cohort member can be indexed within this month-by-month structure. To aid users, a number of variables other than the event history arrays have already been created using the event history format. For example, the month of the respondent's birth and the month of the respondent's interview are both presented in continuous month format. Second, the event history arrays can be constructed using actual month and year dates. Using this approach, the same respondent, born in April 1980, started receiving public assistance in January 1997 and left the program in June 1997.

The data set includes variables for the event history arrays for every month number under the continuous month scheme. However, actual month and year dates are only included for a given respondent in months in which transitions occurred. That is, the respondent in the example above would have continuous month variables relating to program participation status for every month beginning with 172 (the month of the respondent's 14th birthday) but would only have actual date variables for January and June 1997. To aid users in moving between the two dating schemes, table 1 in this appendix contains a crosswalk between actual dates and continuous month numbers.

The continuous month system allows researchers to easily compare the times at which various events occurred in the youth's life. The following example illustrates a case in which a youth turns 17 in February 1997 and, in the same month (month 194 in the continuous timeline), begins receiving AFDC. She remains on AFDC for 2 months (month 194 and month 195). The following table illustrates this information using a month-by-month timeline.

Non-Employment Event History Array Incorporating the Month-by-Month Timeline

Question Name	Participation Status	Month Number (created by the user)	Birth date: CV_CHILD_BIRTH_MONTH
AFDC_STATUS_1997.01	0	193	
AFDC_STATUS_1997.02	1	194	194
AFDC_STATUS_1997.03	1	195	
AFDC_STATUS_1997.04	0	196	

In addition linking the dates through the crosswalk, any user who selects an event array will be able to extract from the CD additional characteristics that relate to that event. The following table presents the additional information that a user may chose to link to the non-employment event history arrays. A user is able to create these and other variable combinations for any status array.

Status	Variable 1	Variable 2	Variable 3
Enrollment	Highest grade attended	Highest grade completed	Highest degree earned
Cohabitation/Marital	Partner's race	Partner's religion	Partner's grade level
AFDC	Received during month	Benefit amount	People benefiting

Continuous Week Timeline

In the event history section of the CD-ROM, an employment history of the number of weeks worked is presented in a continuous week-by-week status array. This array is very similar to the month-by-month scheme used for the rest of the event history variables. In this format, the first week of January 1980 is numbered week 1, the second week of January 1980 is numbered week 2, and so on; weeks are listed by exact date as well. This week-by-week array lists each respondent's employment status for each week since the month of the respondent's 14th birthday.

As with the month-by-month arrays, respondents have data available for every week in the continuous week arrays but only for transition weeks in the actual date arrays. However, users should note that, due to technical considerations, the variable titles in the data number weeks by calendar year rather than from week 1 through week 992. For example, a variable might read "1994 Employment: Employment Status in Week 07" instead of "Employment: Employment Status in Week 737." Table 2 in this appendix provides a crosswalk between the actual beginning date of each week, the continuous week number, and the week number for each week in a calendar year. This table begins with January 1994 because this is the first month in which any NLSY97 respondent could have reached age 14, the youngest age for which the employment event history variables were created. Users who need to see earlier continuous week numbers should contact NLS User Services.

Naming Conventions

The question names of the NLSY97 event history variables incorporate the dates to which the variables apply. The marital status and program participation array titles include the month and year (e.g., AFDC_AMT_1997.12 corresponds to December 1997). Likewise, employment array variables are listed for each week and year (e.g., EMP_STATUS_1997.52 corresponds to the 52nd week in 1997). The schooling variables are yearly and employ a slightly different system. In general, these variables refer to the school year rather than the calendar year. That is, 1991 in a variable title or in the data for a variable generally indicates the school year starting in fall 1991 and ending in spring 1992.

Table 1. Continuous Month Crosswalk

Month and year	Continuous month number	Month and year	Continuous month number
January 1980	1	July 1983	43
February 1980	2	August 1983	44
March 1980	3	September 1983	45
April 1980	4	October 1983	46
May 1980	5	November 1983	47
June 1980	6	December 1983	48
July 1980	7	January 1984	49
August 1980	8	February 1984	50
September 1980	9	March 1984	51
October 1980	10	April 1984	52
November 1980	11	May 1984	53
December 1980	12	June 1984	54
January 1981	13	July 1984	55
February 1981	14	August 1984	56
March 1981	15	September 1984	57
April 1981	16	October 1984	58
May 1981	17	November 1984	59
June 1981	18	December 1984	60
July 1981	19	January 1985	61
August 1981	20	February 1985	62
September 1981	21	March 1985	63
October 1981	22	April 1985	64
November 1981	23	May 1985	65
December 1981	24	June 1985	66
January 1982	25	July 1985	67
February 1982	26	August 1985	68
March 1982	27	September 1985	69
April 1982	28	October 1985	70
May 1982	29	November 1985	71
June 1982	30	December 1985	72
July 1982	31	January 1986	73
August 1982	32	February 1986	74
September 1982	33	March 1986	75
October 1982	34	April 1986	76
November 1982	35	May 1986	77
December 1982	36	June 1986	78
January 1983	37	July 1986	79
February 1983	38	August 1986	80
March 1983	39	September 1986	81
April 1983	40	October 1986	82
May 1983	41	November 1986	83
June 1983	42	December 1986	84

Table 1. Continuous Month Crosswalk (Continued)

Month and year	Continuous month number	Month and year	Continuous month number
January 1987	85	July 1990	127
February 1987	86	August 1990	128
March 1987	87	September 1990	129
April 1987	88	October 1990	130
May 1987	89	November 1990	131
June 1987	90	December 1990	132
July 1987	91	January 1991	133
August 1987	92	February 1991	134
September 1987	93	March 1991	135
October 1987	94	April 1991	136
November 1987	95	May 1991	137
December 1987	96	June 1991	138
January 1988	97	July 1991	139
February 1988	98	August 1991	140
March 1988	99	September 1991	141
April 1988	100	October 1991	142
May 1988	101	November 1991	143
June 1988	102	December 1991	144
July 1988	103	January 1992	145
August 1988	104	February 1992	146
September 1988	105	March 1992	147
October 1988	106	April 1992	148
November 1988	107	May 1992	149
December 1988	108	June 1992	150
January 1989	109	July 1992	151
February 1989	110	August 1992	152
March 1989	111	September 1992	153
April 1989	112	October 1992	154
May 1989	113	November 1992	155
June 1989	114	December 1992	156
July 1989	115	January 1993	157
August 1989	116	February 1993	158
September 1989	117	March 1993	159
October 1989	118	April 1993	160
November 1989	119	May 1993	161
December 1989	120	June 1993	162
January 1990	121	July 1993	163
February 1990	122	August 1993	164
March 1990	123	September 1993	165
April 1990	124	October 1993	166
May 1990	125	November 1993	167
June 1990	126	December 1993	168

Table 1. Continuous Month Crosswalk (Continued)

Month and year	Continuous month number	Month and year	Continuous month number
January 1994	169	August 1997	212
February 1994	170	September 1997	213
March 1994	171	October 1997	214
April 1994	172	November 1997	215
May 1994	173	December 1997	216
June 1994	174	January 1998	217
July 1994	175	February 1998	218
August 1994	176	March 1998	219
September 1994	177	April 1998	220
October 1994	178	May 1998	221
November 1994	179	June 1998	222
December 1994	180	July 1998	223
January 1995	181	August 1998	224
February 1995	182	September 1998	225
March 1995	183	October 1998	226
April 1995	184	November 1998	227
May 1995	185	December 1998	228
June 1995	186	January 1999	229
July 1995	187	February 1999	230
August 1995	188	March 1999	231
September 1995	189	April 1999	232
October 1995	190	May 1999	233
November 1995	191	June 1999	234
December 1995	192	July 1999	235
January 1996	193	August 1999	236
February 1996	194	September 1999	237
March 1996	195	October 1999	238
April 1996	196	November 1999	239
May 1996	197	December 1999	240
June 1996	198	January 2000	241
July 1996	199	February 2000	242
August 1996	200	March 2000	243
September 1996	201	April 2000	244
October 1996	202	May 2000	245
November 1996	203	June 2000	246
December 1996	204	July 2000	247
January 1997	205	August 2000	248
February 1997	206	September 2000	249
March 1997	207	October 2000	250
April 1997	208	November 2000	251
May 1997	209	December 2000	252
June 1997	210	January 2001	253
July 1997	211	February 2001	254

Table 1. Continuous Month Crosswalk (Continued)

Month and year	Continuous month number	Month and year	Continuous month number
March 2001	255	August 2001	260
April 2001	256	September 2001	261
May 2001	257	October 2001	262
June 2001	258	November 2001	263
July 2001	259	December 2001	264

Table 2. Continuous Week Crosswalk

Week start date	Continuous week number	Calendar year week number	Week start date	Continuous week number	Calendar year week number
12/26/1993	731	1	10/30/1994	775	45
01/02/1994	732	2	11/06/1994	776	46
01/09/1994	733	3	11/13/1994	777	47
01/16/1994	734	4	11/20/1994	778	48
01/23/1994	735	5	11/27/1994	779	49
01/30/1994	736	6	12/04/1994	780	50
02/06/1994	737	7	12/11/1994	781	51
02/13/1994	738	8	12/18/1994	782	52
02/20/1994	739	9	12/25/1994	783	53
02/27/1994	740	10	01/01/1995	784	1
03/06/1994	741	11	01/08/1995	785	2
03/13/1994	742	12	01/15/1995	786	3
03/20/1994	743	13	01/22/1995	787	4
03/27/1994	744	14	01/29/1995	788	5
04/03/1994	745	15	02/05/1995	789	6
04/10/1994	746	16	02/12/1995	790	7
04/17/1994	747	17	02/19/1995	791	8
04/24/1994	748	18	02/26/1995	792	9
05/01/1994	749	19	03/05/1995	793	10
05/08/1994	750	20	03/12/1995	794	11
05/15/1994	751	21	03/19/1995	795	12
05/22/1994	752	22	03/26/1995	796	13
05/29/1994	753	23	04/02/1995	797	14
06/05/1994	754	24	04/09/1995	798	15
06/12/1994	755	25	04/16/1995	799	16
06/19/1994	756	26	04/23/1995	800	17
06/26/1994	757	27	04/30/1995	801	18
07/03/1994	758	28	05/07/1995	802	19
07/10/1994	759	29	05/14/1995	803	20
07/17/1994	760	30	05/21/1995	804	21
07/24/1994	761	31	05/28/1995	805	22
07/31/1994	762	32	06/04/1995	806	23
08/07/1994	763	33	06/11/1995	807	24
08/14/1994	764	34	06/18/1995	808	25
08/21/1994	765	35	06/25/1995	809	26
08/28/1994	766	36	07/02/1995	810	27
09/04/1994	767	37	07/09/1995	811	28
09/11/1994	768	38	07/16/1995	812	29
09/18/1994	769	39	07/23/1995	813	30
09/25/1994	770	40	07/30/1995	814	31
10/02/1994	771	41	08/06/1995	815	32
10/09/1994	772	42	08/13/1995	816	33
10/16/1994	773	43	08/20/1995	817	34
10/23/1994	774	44	08/27/1995	818	35

Table 2. Continuous Week Crosswalk (Continued)

Week start date	Continuous week number	Calendar year week number	Week start date	Continuous week number	Calendar year week number
09/03/1995	819	36	07/07/1996	863	28
09/10/1995	820	37	07/14/1996	864	29
09/17/1995	821	38	07/21/1996	865	30
09/24/1995	822	39	07/28/1996	866	31
10/01/1995	823	40	08/04/1996	867	32
10/08/1995	824	41	08/11/1996	868	33
10/15/1995	825	42	08/18/1996	869	34
10/22/1995	826	43	08/25/1996	870	35
10/29/1995	827	44	09/01/1996	871	36
11/05/1995	828	45	09/08/1996	872	37
11/12/1995	829	46	09/15/1996	873	38
11/19/1995	830	47	09/22/1996	874	39
11/26/1995	831	48	09/29/1996	875	40
12/03/1995	832	49	10/06/1996	876	41
12/10/1995	833	50	10/13/1996	877	42
12/17/1995	834	51	10/20/1996	878	43
12/24/1995	835	52	10/27/1996	879	44
12/31/1995	836	1	11/03/1996	880	45
01/07/1996	837	2	11/10/1996	881	46
01/14/1996	838	3	11/17/1996	882	47
01/21/1996	839	4	11/24/1996	883	48
01/28/1996	840	5	12/01/1996	884	49
02/04/1996	841	6	12/08/1996	885	50
02/11/1996	842	7	12/15/1996	886	51
02/18/1996	843	8	12/22/1996	887	52
02/25/1996	844	9	12/29/1996	888	1
03/03/1996	845	10	01/05/1997	889	2
03/10/1996	846	11	01/12/1997	890	3
03/17/1996	847	12	01/19/1997	891	4
03/24/1996	848	13	01/26/1997	892	5
03/31/1996	849	14	02/02/1997	893	6
04/07/1996	850	15	02/09/1997	894	7
04/14/1996	851	16	02/16/1997	895	8
04/21/1996	852	17	02/23/1997	896	9
04/28/1996	853	18	03/02/1997	897	10
05/05/1996	854	19	03/09/1997	898	11
05/12/1996	855	20	03/16/1997	899	12
05/19/1996	856	21	03/23/1997	900	13
05/26/1996	857	22	03/30/1997	901	14
06/02/1996	858	23	04/06/1997	902	15
06/09/1996	859	24	04/13/1997	903	16
06/16/1996	860	25	04/20/1997	904	17
06/23/1996	861	26	04/27/1997	905	18
06/30/1996	862	27	05/04/1997	906	19

Table 2. Continuous Week Crosswalk (Continued)

Week start date	Continuous week number	Calendar year week number	Week start date	Continuous week number	Calendar year week number
05/11/1997	907	20	03/08/1998	950	11
05/18/1997	908	21	03/15/1998	951	12
05/25/1997	909	22	03/22/1998	952	13
06/01/1997	910	23	03/29/1998	953	14
06/08/1997	911	24	04/05/1998	954	15
06/15/1997	912	25	04/12/1998	955	16
06/22/1997	913	26	04/19/1998	956	17
06/29/1997	914	27	04/26/1998	957	18
07/06/1997	915	28	05/03/1998	958	19
07/13/1997	916	29	05/10/1998	959	20
07/20/1997	917	30	05/17/1998	960	21
07/27/1997	918	31	05/24/1998	961	22
08/03/1997	919	32	05/31/1998	962	23
08/10/1997	920	33	06/07/1998	963	24
08/17/1997	921	34	06/14/1998	964	25
08/24/1997	922	35	06/21/1998	965	26
08/31/1997	923	36	06/28/1998	966	27
09/07/1997	924	37	07/05/1998	967	28
09/14/1997	925	38	07/12/1998	968	29
09/21/1997	926	39	07/19/1998	969	30
09/28/1997	927	40	07/26/1998	970	31
10/05/1997	928	41	08/02/1998	971	32
10/12/1997	929	42	08/09/1998	972	33
10/19/1997	930	43	08/16/1998	973	34
10/26/1997	931	44	08/23/1998	974	35
11/02/1997	932	45	08/30/1998	975	36
11/09/1997	933	46	09/06/1998	976	37
11/16/1997	934	47	09/13/1998	977	38
11/23/1997	935	48	09/20/1998	978	39
11/30/1997	936	49	09/27/1998	979	40
12/07/1997	937	50	10/04/1998	980	41
12/14/1997	938	51	10/11/1998	981	42
12/21/1997	939	52	10/18/1998	982	43
12/28/1997	940	1	10/25/1998	983	44
01/04/1998	941	2	11/01/1998	984	45
01/11/1998	942	3	11/08/1998	985	46
01/18/1998	943	4	11/15/1998	986	47
01/25/1998	944	5	11/22/1998	987	48
02/01/1998	945	6	11/29/1998	988	49
02/08/1998	946	7	12/06/1998	989	50
02/15/1998	947	8	12/13/1998	990	51
02/22/1998	948	9	12/20/1998	991	52
03/01/1998	949	10	12/27/1998	992	1

Table 2. Continuous Week Crosswalk (Continued)

Week start date	Continuous week number	Calendar year week number	Week start date	Continuous week number	Calendar year week number
1/3/1999	993	2	11/7/1999	1037	46
1/10/1999	994	3	11/14/1999	1038	47
1/17/1999	995	4	11/21/1999	1039	48
1/24/1999	996	5	11/28/1999	1040	49
1/31/1999	997	6	12/5/1999	1041	50
2/7/1999	998	7	12/12/1999	1042	51
2/14/1999	999	8	12/19/1999	1043	52
2/21/1999	1000	9	12/26/1999	1044	53
2/28/1999	1001	10	1/2/2000	1045	1
3/7/1999	1002	11	1/9/2000	1046	2
3/14/1999	1003	12	1/16/2000	1047	3
3/21/1999	1004	13	1/23/2000	1048	4
3/28/1999	1005	14	1/30/2000	1049	5
4/4/1999	1006	15	2/6/2000	1050	6
4/11/1999	1007	16	2/13/2000	1051	7
4/18/1999	1008	17	2/20/2000	1052	8
4/25/1999	1009	18	2/27/2000	1053	9
5/2/1999	1010	19	3/5/2000	1054	10
5/199/1999	1011	20	3/12/2000	1055	11
5/16/1999	1012	21	3/19/2000	1056	12
5/23/1999	1013	22	3/26/2000	1057	13
5/30/1999	1014	23	4/2/2000	1058	14
6/6/1999	1015	24	4/9/2000	1059	15
6/13/1999	1016	25	4/16/2000	1060	16
6/20/1999	1017	26	4/23/2000	1061	17
6/27/1999	1018	27	4/30/2000	1062	18
7/4/1999	1019	28	5/7/2000	1063	19
7/11/1999	1020	29	5/14/2000	1064	20
7/18/1999	1021	30	5/21/2000	1065	21
7/25/1999	1022	31	5/28/2000	1066	22
8/1/1999	1023	32	6/4/2000	1067	23
8/8/1999	1024	33	6/11/2000	1068	24
8/15/1999	1025	34	6/18/2000	1069	25
8/22/1999	1026	35	6/25/2000	1070	26
8/29/1999	1027	36	7/2/2000	1071	27
9/5/1999	1028	37	7/9/2000	1072	28
9/12/1999	1029	38	7/16/2000	1073	29
9/19/1999	1030	39	7/23/2000	1074	30
9/26/1999	1031	40	7/30/2000	1075	31
10/3/1999	1032	41	8/6/2000	1076	32
10/10/1999	1033	42	8/13/2000	1077	33
10/17/1999	1034	43	8/20/2000	1078	34
10/24/1999	1035	44	8/27/2000	1079	35
10/31/1999	1036	45	9/3/2000	1080	36

Table 2. Continuous Week Crosswalk (Continued)

Week start date	Continuous week number	Calendar year week number	Week start date	Continuous week number	Calendar year week number
9/10/2000	1081	37	5/13/2001	1116	29
9/17/2000	1082	38	5/20/2001	1117	20
9/24/2000	1083	39	5/20/2001	1118	21
10/1/2000	1084	40	5/27/2001	1119	22
10/8/2000	1085	41	6/3/2001	1120	23
10/15/2000	1086	42	6/10/2001	1121	24
10/22/2000	1087	43	6/17/2001	1122	25
10/29/2000	1088	44	6/24/2001	1123	26
11/5/2000	1089	45	7/1/2001	1124	27
11/12/2000	1090	46	7/8/2001	1125	28
11/19/2000	1091	47	7/15/2001	1126	29
11/26/2000	1092	48	7/22/2001	1127	30
12/3/2000	1093	49	7/29/2001	1128	31
12/10/2000	1094	50	8/5/2001	1129	32
12/17/2000	1095	51	8/12/2001	1130	33
12/24/2000	1096	52	8/19/2001	1131	34
12/31/2000	1097	53	8/26/2001	1132	35
1/7/2001	1098	1	9/2/2001	1133	36
1/14/2001	1099	2	9/9/2001	1134	37
1/21/2001	1100	3	9/16/2001	1135	38
1/28/2001	1101	4	9/23/2001	1136	39
2/4/2001	1102	5	9/30/2001	1137	40
2/11/2001	1103	6	10/7/2001	1138	41
2/18/2001	1104	7	10/14/2001	1139	42
2/25/2001	1105	8	10/21/2001	1140	43
3/4/2001	1106	9	10/28/2001	1141	44
3/11/2001	1107	10	11/4/2001	1142	45
3/18/2001	1108	11	11/11/2001	1143	46
3/25/2001	1109	12	11/18/2001	1144	47
4/1/2001	1110	13	11/25/2001	1145	48
4/8/2001	1111	14	12/2/2001	1146	49
4/15/2001	1112	15	12/9/2001	1147	50
4/22/2001	1113	16	12/16/2001	1148	51
4/29/2001	1114	17	12/23/2001	1149	52
5/6/2001	1115	18	12/30/2001	1150	53

NLSY97 Appendix 8:

Instrument Rosters

The NLSY97 uses rosters in various sections in which information is collected on a number of persons, schools, or employers. Rosters are an important part of the NLSY97 data set. These grids of information help researchers to analyze data in an efficient and accurate way. However, the structure and use of rosters may be somewhat confusing, so it is vital that researchers understand how they are constructed. After describing the construction and use of rosters, the following pages list the variable names, titles, and reference numbers for the various instrument rosters used during the round 1–4 interviews. These lists are intended to aid researchers in identifying the types of information organized in each roster and to better follow the flow of information through the interview.

WHAT IS A ROSTER?

A roster may be thought of as a list—for example, a list of household members, a list of employers, or a list of children. A respondent with two children will have data on the first two lines of the child list, or child roster. A respondent with four employers will have information on the first four lines of the employer roster. In addition to the name of the person or item (which is not released to the public), the roster contains other basic information, such as the age, race, and labor force status of household members or the start date and stop date for each employer.

In the paper-and-pencil interviews (PAPI) of older NLS cohorts, the questionnaires included a chart or grid listing this type of information, like the one shown in Figure 1 below. For example, in the household roster grid, each household member's name was entered in a separate row. The interviewer asked the respondent for each member's date of birth, enrollment status, employment status, etc., filling in the answers in the appropriate column. This completed household roster contained all the pertinent information about household residents, and researchers could easily use the variables based on this roster to examine characteristics of household members.

Figure 1. Sample PAPI Roster Grid

What are the names of all family members who are living in your home?

Name	What is __'s relationship to you?	How old is __ today?	(Age 4 and older) Is __ enrolled in school?	(Age 16 and older)		
				How many weeks did __ work in the last 12 months?	How many hours did __ usually work per week?	What kind of work was __ doing in the past 12 months?
Susan	Mother	45	No	50	25	Graphic design
John	Father	49	No	50	40	Banking
Jimmy	Brother	17	Yes	35	15	Food service
Sally	Sister	12	Yes	(n/a)	(n/a)	(n/a)
Robert	Brother	3	(n/a)	(n/a)	(n/a)	(n/a)
Jane	Grandmother	77	No	0	(n/a)	(n/a)

When the NLS surveys changed to computer-assisted personal interviewing (CAPI), rosters became a very important way of organizing information during the interview. Instead of using an actual grid, however, CAPI questionnaires include a series of questions that gather the same types of information that would have been included in the grid in a paper-and-pencil interview. The computer then moves the answers to these questions into a grid, creating a roster from the information.

After the roster is created, it can be used to guide subsequent portions of the interview. For example, during the interview the NLSY97 questionnaire gathers the names, dates of attendance, and level of school (secondary

school or college) for each of the respondent's schools and organizes them into a roster. The rest of the school section then asks questions about the first school on the roster, followed by questions about the second school, then the third, and so on. The information about the level of the school determines whether the respondent is asked questions that apply to high school or college.

The information from the roster is also presented on the CD as an organized list of data, so that these variables are easy for researchers to access. To the user, the school roster appears as a consolidated block of variables that contains key information such as dates of enrollment, an identification number for the school, and variables indicating the type (private or public) and level (junior high, high school, college) of the school. For example, the variables in the round 2 school roster are listed in Figure 2, along with their reference numbers. Thus, rosters are a way of organizing information both for researchers and for the actual interview so that questions are asked in a logical manner.

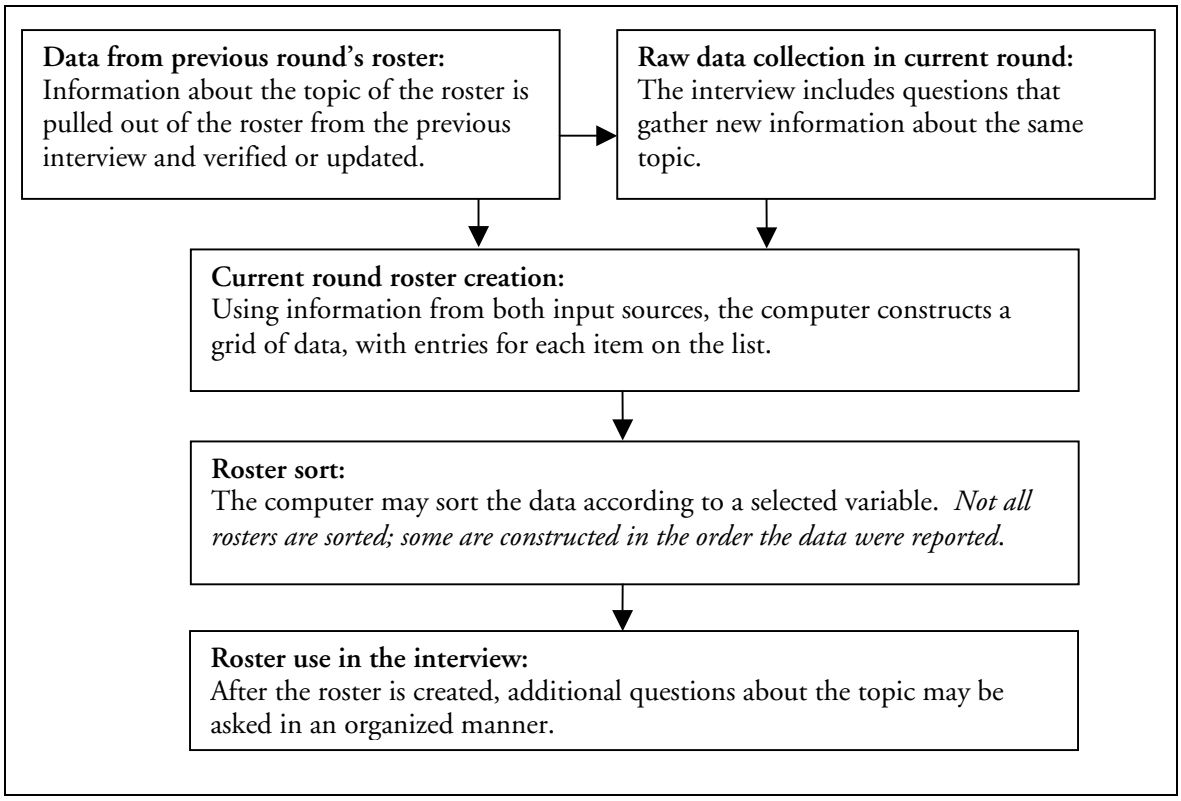
Figure 2. List of Variables in CAPI-Generated School Roster

Question Name	Variable Title	Reference Numbers (one for each school)
NEWSCHOOL_PERIODS.xx	Number of Times R Enrolled in School xx	R24605.–R24610.
NEWSCHOOL_START1.xx	Month/Year R Start 1st Enrollment in School xx	R24611.00–R24616.01
NEWSCHOOL_START2.xx	Month/Year R Start 2nd Enrollment in School xx	R24617.00–R24620.01
NEWSCHOOL_START3.xx	Month/Year R Start 3rd Enrollment in School xx	R24621.00–R24621.01
NEWSCHOOL_STOP1.xx	Month/Year R End 1st Enrollment in School xx	R24622.00–R24627.01
NEWSCHOOL_STOP2.xx	Month/Year R End 2nd Enrollment in School xx	R24628.00–R24631.01
NEWSCHOOL_STOP3.xx	Month/Year R End 3rd Enrollment in School xx	R24632.00–R24632.01
NEWSCHOOL_SCHCODE.xx	School Code Elementary, Middle, High, College	R24633.–R24638.
NEWSCHOOL_INTERVIEW.xx	Which Survey Round School xx Reported in	R24639.–R24644.
NEWSCHOOL_TYPE.xx	Type of School xx R has Attended	R24645.–R24650.
NEWSCHOOL_PUBID.xx	PUBID of School xx R has Attended	R24651.–R24656.

HOW ARE ROSTERS CREATED DURING THE INTERVIEW?

This section outlines the process used during the interview to create a roster. Rosters may include data from both previous interviews and the current interview. After the roster is created and sorted, it can be used to guide the rest of the interview. Figure 3 provides a pictorial overview of the creation of a roster.

Figure 3. How Rosters Are Created



Data from previous interviews: As shown in the figure, creation of a roster for the current round often begins with information found in the roster from the previous round. The appropriate respondent-specific data are saved on the interviewer's laptop before he or she administers the survey. When the interview gets to a point where roster information is collected, the data from the previous round's roster are often used as the base for the current roster. The respondent verifies and updates the information. If no changes have occurred since the last interview—for example, if exactly the same people live in the respondent's household—then the current round's roster will be the same as the one from the previous round.

For example, the interviewer reads a list of all of the people on the household roster from the last interview. The respondent first states whether any of those people have moved out of the household and then reports new household members. If any members remain from the previous year, their information—date of birth, gender, race/ethnicity, etc.—is carried over from the previous interview, and any missing data are collected. This method is more efficient than asking the respondent to report all household members every year.

Raw data collection: After the respondent and interviewer review and update the roster from the previous round, the survey collects current information. For example, new people might have moved into the household, so the interviewer asks the respondent about their characteristics. At this point, the respondent is done answering questions that will fill up the data grid on a particular topic.

Roster creation and roster sort: Using the updated roster from the previous round and the new raw data just collected, the computer creates a new roster for the current round. For example, the employer roster contains the following information for each job: a unique identification number for the employer, employment dates, whether the job was current at the interview date, whether the job was in the military, and whether the job was an internship. If the respondent had held the job at the time of the previous interview, the start date and employer identification number are carried over from the old roster, and the other information is taken from

the questions at the beginning of the employment section for the current year. Similarly, the household roster contains information from the previous interview about household members reported at that time and data from the current interview about new household members.

In some cases, the computer also sorts the roster and puts the items in order based on a specified variable. For example, in the round 1 household roster, all youths in the age range of the NLSY97 cohort were listed first, and then all other household members were listed from oldest to youngest. The employer roster is sorted by job end date so that the most recent jobs are listed first.

Roster use in the interview: Finally, the roster is used to determine the order in which the other questions about each topic are asked. In most cases, the survey collects far more information than is stored in the actual roster, and the answers to these questions remain outside the roster as raw data. So that the interview makes sense to the respondent, these additional questions are asked about the people or things on the roster in the order that the people or things are listed.

For example, the respondent first answers questions about industry, occupation, rate of pay, etc., for the first employer listed on the roster. The same questions are then asked about the second job, then the third job, and so on. Similarly, the first set of questions about household members refers to the first person listed on the roster. When all of those questions have been answered, the same questions are asked about the second person, the third person, etc.

Example—Creation and use of the employer roster
The following example illustrates the structure and use of the employer roster. Most aspects of this example apply to other NLSY97 rosters as well.

Roster Creation in Round 1
Raw data collection: The round 1 survey asked for the names of all employers for whom the respondent had worked since age 14. Assume that a respondent named Emma reported delivering the *Smalltown Press* when she was 14, then switching companies and delivering the *County Register*, and finally working in her parents' business, Peel's Corner Store, at the time of the round 1 interview. For this example, the newspaper delivery jobs are assumed to be employee jobs and not freelance-type work. The survey then assigned a unique identification number (UID) in the order the jobs were reported: 9701 for the *Smalltown Press*, 9702 for the *County Register*, and 9703 for Peel's Store.

Roster creation and roster sort: After the UIDs were assigned, Emma reported the dates she started and stopped working for each employer. At this point, the survey program sorted the jobs according to stop date, so that the most recent employer was employer #01, the next most recent was employer #02, and so on. Therefore, Peel's Store (UID 9703) became job #01 on the roster, the *County Register* (UID 9702) was listed as job #02, and the *Smalltown Press* (UID 9701) was listed third. Key information about each employer, including the unique ID number and dates of employment, was organized in the employer roster. All of the information about Peel's Store is located in variables numbered #01 in the title, the *County Register* data are in variables numbered #02, and so on.

Example: Emma's Round 1 YEMP Roster

Employer	UID	Round 1 Roster Line #
Smalltown Press	9701	03
County Register	9702	02
Peel's Store	9703	01

Roster use in the interview: Throughout the rest of the employment section, the employer numbers remain constant, so that each variable containing, for example, the phrase "Job #03" or "Employer #03" refers to Emma's *Smalltown Press* job. Note that the *Smalltown Press* is **not** the third employer Emma reported at the

beginning of the employment section of the interview. It became employer #03 during the roster sort because the other two jobs were more recent.

Roster Creation in Round 2

Data from previous interviews: The employer information was collected in a similar manner in subsequent rounds. Because data were available from the previous interview, they could be used in the construction of the round 2 roster. Before the survey was fielded, survey staff loaded information about each respondent into the interviewers' laptops. In Emma's case, part of this information would be the list of employers she reported in round 1.

Raw data collection: During the survey, respondents first provided information about employers who were current at the last interview date. Assume that Emma stated that she worked at Peel's Store for several months after the round 1 interview. Respondents next reported new employers since the last interview date in no particular order. Emma reported only one additional job, waiting tables at Steed's Diner after she turned 16. At this point UIDs were given to each employer. Because Peel's Store was previously reported, it already had a UID—9703—assigned during the last interview. Steed's Diner was a new employer in round 2, so it was given a UID of 9801.

Roster creation and roster sort: Emma then reported the date she stopped working at each job, and the roster was sorted according to these stop dates. At the round 2 interview, the diner job was more recent, so it was listed as job #01 on the roster, and the store became job #02. At this point, the roster contains information from multiple survey rounds. The UID and start date of the Peel's Store job are carried over from round 1, while the stop date of the store job and all the information about Steed's Diner comes from round 2. Because Emma had not worked for the *Smalltown Press* or the *County Register* since the round 1 interview, neither of those employers is listed on the round 2 roster.

Example: Emma's Round 2 YEMP Roster

Employer	UID	Round 1 Roster Line #	Round 2 Roster Line #
Peel's Store	9703	01	02
Steed's Diner	9801	—	01

Roster use in the interview: Just as in round 1, the employer numbers remain the same for the rest of the interview. As Emma answered questions about Steed's Diner, her rate of pay, hours worked, etc., were recorded in the "Employer #01" questions. Peel's Store data were recorded in the "Employer #2" series.

Roster Creation in Round 3 and Subsequent Surveys

Data from previous interviews: This is used in each round as it was in round 2. Data from earlier interviews is loaded into the interviewer's laptop before the survey begins so that it can be used in the current survey if applicable.

Raw data collection: During the survey, respondents first provide information about employers who were current at the last interview date. In response to these questions in round 3, Emma reported her ongoing employment at Steed's Diner, where she had been working in round 2. Respondents next report new employers since the last interview date in no particular order; this series includes a check of employers from prior to the date of the last interview to determine if the respondent has returned to a previous employer. At this point in round 3, Emma reported that she went back to work at the *Smalltown Press* for a 6-month period between rounds 2 and 3. Both employers retained their original ID numbers, 9801 for the diner and 9701 for the newspaper, despite the break in Emma's employment at the latter.

Roster creation and roster sort: The roster is again sorted according to the stop date of each job. As Emma's current employer, Steed's Diner is listed as job #01 on the round 3 roster. The Smalltown Press becomes job

#02 since Emma left that job before the round 3 interview date. Jobs where employment ended before round 2 are not listed on the round 3 roster..

Example: Emma's Round 3 YEMP Roster

Employer	UID	Round 1 Roster Line #	Round 2 Roster Line #	Round 3 Roster Line #
Steed's Diner	9801	—	01	01
Smalltown Press	9701	03	—	02

Roster use in the interview: The roster line numbers are then used for the duration of the interview, as described above for rounds 1 and 2. In the example, information about Emma's employment at the diner is recorded in the "Employer #01" questions, while data about the newspaper are recorded in the "Employer #02" series..

User Notes: The transition from the freelance section to the regular employee job section (see the introduction to appendix 2, "Employment Variable Creation") creates a disconnect in unique ID codes. If a respondent has a freelance job at age 16 with enough earnings to qualify as self-employed, and continues that job past his or her 18th birthday, the job will now appear on the regular employer roster rather than the freelance roster. When the job was in the freelance section, it was assigned a freelance UID. However, these UIDs cannot be transferred to the regular employer roster because the numbering system is different. On the other hand, it is misleading to assign a round 4 UID, when the self-employed job first appears on the regular employer roster, because the job was preexisting. So, to indicate that a self-employed job existed during a previous interview, it is assigned a UID of 199999 on the round 4 roster.

If a self-employed job is new in the round 4 interview—it had never been reported in the freelance section—it will be assigned a regular round 4 UID.

HOW SHOULD RESEARCHERS USE THE ROSTER DATA IN ANALYSIS?

The data set is organized so that rosters can easily be found and used in research. Because rosters present key pieces of information in a structured format, they are the best place to obtain that information. All variables found on rosters have "Roster Item" as their main area of interest. Each roster has a unique name that serves as the beginning of the question name for all variables on the roster; the same name appears at the beginning of the variable title for each item on the roster. Different rosters have been used in different rounds, depending on the topics included in the interview and the type of information collected. The roster names and question names are shown in Figure 4.

Figure 4. Rosters Included Each Round

Roster	Question name	Round 1	Round 2	Round 3	Round 4
Household Information	HHI2 (rd. 1), HHI (rds. 2–4)	✓	✓	✓	✓
Nonresident Roster	NONHHI	✓	✓	✓	✓
Youth Information	YOUTH	✓			
School Roster	NEWSCHOOL		✓	✓	✓
Employer Roster	YEMP	✓	✓	✓	✓
Freelance Jobs Roster	FREELANCE		✓	✓	✓
Training Roster	TRAINING			✓	✓
Biological Children Roster	BIOCHILD	✓	✓	✓	✓
Parent Household Information	PARHHI	✓			
Parent Youth Information	PARYOUTH	✓			

***Data hint*➔**	Researchers can locate rosters on the data CD-ROM by looking at the roster item area of interest, by selecting the appropriate question name, or by searching the any word in context index for variables with "ros" or "roster" and the name of the roster of interest in the title.

User Notes: When the NLSY97 data set was initially created, variables could only be assigned to one area of interest. The newer data extraction software permits variables to be linked to multiple areas of interest. However, additional areas have not been assigned to every variable. Because roster items were initially located in the roster item area of interest on the CD, they may not be grouped with the rest of the data on a particular topic. For example, the school roster variables may not appear if the user searches for the "School Experience" area of interest. For this reason, it is very important that users become familiar with the rosters used in the data set. If a roster is available on the topic of a particular research project, users should always locate that roster using one of the search techniques mentioned above and examine it before using the other (non-roster) variables that relate to their research.

Using rosters in single-round analyses: When looking at the data set, users will notice that many questions are repeated for each person or thing on the roster, and the titles for these repeated questions include a number. This number indicates the line on the roster that corresponds to the person or item being described in that variable. For example, the question "Self-Employed Business/Industry Job 02" indicates the industry of the second job listed on the respondent's self-employment roster. The researcher may then want to examine information such as the respondent's start and stop dates or rate of pay for that job. To find this information, he or she can then look at the data for those items contained in the roster for job #02, or the self-employment job that is on the second line of the roster. For all other questions asked after the roster was created in that same survey year, job #02 will refer to the same self-employment job.

Users should be aware that, in some cases, the information contained in the rosters actually appears in the data set more than once. As Figure 1 suggests, data may first be included at the point in the interview when the information was actually collected. For example, the round 1 screener question SE-28 asked the household informant for the date of birth of each household member. After all the raw data had been gathered, the computer sorted all the answers and created the household roster. At this point the date of birth information is also located in the round 1 roster variables named HHI2_DOB. In the case of the round 1 household roster, both the raw data and roster items are included in the data set.

In other cases, the raw answers may be blanked out of the public use data set. If a reference number is not listed for a given question in the questionnaire, then that raw data item may only be represented in roster

form. For example, answers to the raw data questions used to create the employer roster are blanked out and do not appear on the CD. In the printed questionnaire, these questions have no reference numbers. However, all of the data collected in these questions (except for confidential information like the name of the employer) appears in the employer roster.

> *User Notes:* Even though the data may appear more than once, **survey staff strongly recommend that researchers use the roster information rather than the raw data whenever possible.** Survey staff are working to eliminate these duplicate sources of information.

For some variables, the roster information may be more accurate because some rosters are updated during the interview if the initial report was inaccurate. When survey staff prepare the data for release, they clean up the rosters if necessary but do not necessarily clean the corresponding raw data. Finally, because many rosters are sorted in a particular order, the number of a person or item on the roster will not match the number in the questions that precede roster creation. For example, in the household screener (the SE questions), person #01 is the first household resident mentioned to the interviewer. In the household roster and all later interview questions, person #01 is the oldest person in the household who was eligible for the NLSY97. Person #01 in the SE questions might be person #05 on the roster. It can be very difficult to determine to which person, school, or job a pre-sort question refers. For all of these reasons, roster data are always preferable to raw data in cases where both are available.

Using rosters from more than one round: Because the NLSY97 is a longitudinal survey, researchers often want to link data across survey rounds. However, household residents, jobs, and so on may move around on the roster in different interviews. That is, a father who was listed third on the roster in round 1 might move to position 2 or 4 in round 2. The unique identification numbers (UIDs) are the key to finding the same person or thing in different rounds. Most of the rosters contain variables assigning a unique number to each person or thing listed. This number never changes and can be used to link roster items across rounds. In some cases, it also makes it possible to link people between two different rosters in the same survey. For example, beginning in round 2 the unique ID listed for a child on the biological children roster is the same one assigned to that child on the household roster. Researchers can therefore examine data on both rosters about the same child.

An additional feature of most unique ID numbers is that they incorporate an indicator of the round in which the person or item was first reported. For example, IDs of roster items reported in round 1 may begin with "1" or "97," while those first reported in round 2 begin with "2" or "98." (Beginning with round 3, 4-digit years are used so that IDs begin with "1999" rather than just "99.")

Example—Use of the employer roster in analysis

Continuing the above example, this section explains how to use rosters in data analyses. Although the employer roster is used in the example, most aspects apply to other NLSY97 rosters as well. Emma's information, as organized in the employer rosters, can be used to examine the characteristics of her jobs at the date of each interview or over time. This example focuses primarily on the round 2 employer roster, but use of the roster is similar in each subsequent round.

As described above, Emma worked for Peel's Store and Steed's Diner during the period between the round 1 and round 2 interviews. Information about these employers was sorted and a roster constructed with the most recent employer appearing first. A researcher using these data would need to be aware of the impact of roster construction.

Because the roster is sorted and employers reported in different rounds may be mixed, variables with "Employer #01" in the title do not necessarily refer to employer number 9701, 9801, etc. The #01 refers solely to the order of the job as listed on the current year's roster. The unique identification numbers provide

a crosswalk between the two systems of identification. The UIDs also allow users to link employers across survey rounds and to identify the round in which an employer was first reported.

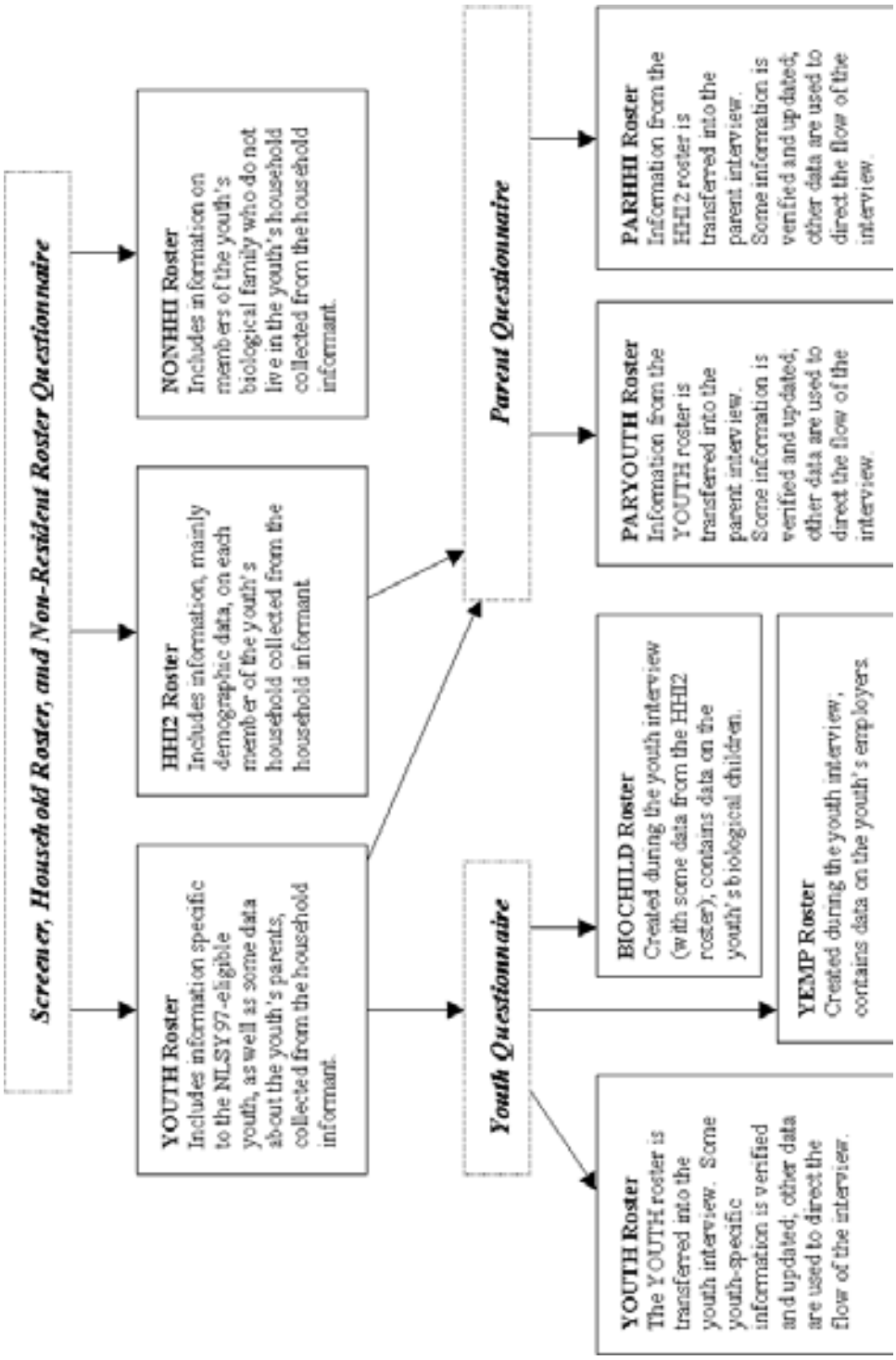
For example, Emma's value for the round 2 variable R24761., "YEMP, Employer 02 Unique ID (Ros Item)," would be 9703—Peel's Store. The user can identify this as an ID assigned in round 1 because it starts with "97," and look at the round 1 UID variables (R05311.–R05317.) to match the employer. In Emma's case, the variable for employer #01 in round 1 would have UID 9703. Therefore, the researcher knows that information about employer #01 in round 1 refers to the same job as variables about employer #02 in round 2. The variables from the two rounds can then be compared to determine if there were any changes in characteristics such as hours worked, rate of pay, occupation, etc.

The roster line numbers and UID variables in the event history data work in the same way. For example, a researcher might want to know Emma's employment status in the first and last week of 1998. In the first week of 1998 (variable EMP_STATUS.01.98), Emma was working at her parents' store, so the status variable would have a value of 9703. Using this UID, researchers can link that job to all of the other information collected during the interview. For the last week of 1998 (variable EMP_STATUS.01.98), when Emma was working at Steed's Diner, the status variable would have a value of 9801. The second set of event history variables, the start and stop dates of each job, uses the roster line numbers. For these variables, the number in the variable title refers to the same job as in the main data set. For example, in the main round 2 data Peel's Store is job #02. The start and stop dates for Peel's Store in the event history data (variables EMP_START_WEEK.02 and EMP_END_WEEK.02) will also have #02 in the variable title.

SPECIAL INFORMATION ABOUT ROUND 1 ROSTERS

In round 1, a number of rosters were used to organize information from the Screener, Household Roster, and Non-Resident Roster Questionnaire. Some of this information was then transferred into the parent and youth interviews for verification and for use in determining question paths. Figure 5 identifies the key rosters in the round 1 survey and shows how they were used in different parts of the survey.

Figure 5. Construction and Use of Round 1 Rosters



ROUND 1

HHI2 Roster (Household enumeration)
PREFIX = "HHI2" (e.g., HHI2_AGE.01)

Variable Name	Title: All end in (Scr Ros Item)	Reference #
_AGE.xx	Age of HH Member xx as of Intdate	R10803.–R10818.
_ASVAB.xx	Was HH Member xx Flagged for ASVAB	R10819.–R10834.
_DOB.xx	Date of Birth of HH Member xx	R10851.01–R10865.02
_DADID.xx	ID Number of HH Member xx Bio Dad	R10835.–R10850.
_EMPLOYED.xx	Employment Status of HH Member xx	R10898.–R10913.
_ENROLLNEXT.xx	Will HH Member xx Be Enrolled Next Fall	R10914.–R10929.
_ENROLLSTAT.xx	Is HH Member xx Currently Enrolled	R10930.–R10945.
_ETHNICITY.xx	Is HH Member xx Hispanic?	R10946.–R10961.
_HHI1ID.xx	ID of HH Member xx from HHI1 Roster	R10978.–R10993.
_HIGHGRADE.xx	HH Member xx Highest Grade Completed	R10994.–R11009.
_HHID.xx	HH Member xx ID	R11010.–R11025.
_INFORMANT.xx	Is HH Member xx the Informant	R11026.–R11041.
_MARSTAT.xx	HH Member xx Marital Status	R11042.–R11057.
_MOMID.xx	HH Member xx Bio Moms ID	R11058.–R11073.
_PARTNER.xx	HH Member xx Have a Partner	R11122.–R11137.
_PARTNERID.xx	ID of HH Member xx Partner	R11138.–R11153.
_RACE.xx	Race of HH Member xx	R11154.–R11169.
_REL1.xx	Relationship of Person 1 to HH Member xx	R11170.–R11185.
_REL2.xx	Relationship of Person 2 to HH Member xx	R11319.–R11334.
_REL3.xx	Relationship of Person 3 to HH Member xx	R11342.–R11357.
_REL4.xx	Relationship of Person 4 to HH Member xx	R11358.–R11373.
_REL5.xx	Relationship of Person 5 to HH Member xx	R11374.–R11389.
_REL6.xx	Relationship of Person 6 to HH Member xx	R11390.–R11405.
_REL7.xx	Relationship of Person 7 to HH Member xx	R11406.–R11421.
_REL8.xx	Relationship of Person 8 to HH Member xx	R11422.–R11437.
_REL9.xx	Relationship of Person 9 to HH Member xx	R11438.–R11453.
_REL10.xx	Relationship of Person 10 to HH Member xx	R11186.–R11201.
_REL11.xx	Relationship of Person 11 to HH Member xx	R11202.–R11217.
_REL12.xx	Relationship of Person 12 to HH Member xx	R11218.–R11233.
_REL13.xx	Relationship of Person 13 to HH Member xx	R11234.–R11249.
_REL14.xx	Relationship of Person 14 to HH Member xx	R11250.–R11265.
_REL15.xx	Relationship of Person 15 to HH Member xx	R11266.–R11281.
_REL16.xx	Relationship of Person 16 to HH Member xx	R11282.–R11297.
_REVDOLEL.xx	Is HH Member xx DOL Eligible (Revised)	R11454.–R11469.
_REVETPEL.xx	Is HH Member xx ETP Eligible (Revised)	R11470.–R11485.
_REVSTPEL.xx	Is HH Member xx STP Eligible (Revised)	R11486.–R11501.
_SEX.xx	Gender of HH Member xx	R11502.–R11517.
_SPECIAL.xx	HH Member xx Approved Special Accommodations	R11518.–R11533.
_SPOPARID.xx	ID of HH Member xx Spouse or Partner	R11541.–R11556.
_SPOUSEID.xx	ID of HH Member xx Spouse in HH	R11557.–R11572.
_UID.xx	HH Member xx Unique ID	R11621.–R11636.

NONHHI Roster (Non-resident relative enumeration)
PREFIX = "NONHHI" (e.g., NONHHI_AGE.01)

Variable Name	Title: All end in (Scr Ros Item)	Reference #
_AGE.xx	Age Non-Res Member xx at Interview Date	R11637.–R11659.
_DECEASED.xx	Is Non-Res Member xx Deceased	R11660.–R11682.
_DEGREE.xx	Non-Res Member xx Have a Degree	R11683.–R11703.
_EMPLOYED.xx	Employment Status of Non-Res Member xx	R11704.–R11724.
_ETHNICITY.xx	Is Non-Res Member xx Hispanic	R11725.–R11745.
_FLAG.XX	How Is Non-Res Member xx Flagged	R11746.–R11768.
_HIGHGRADE.xx	HGC by Non-Res Member xx	R11769.–R11789.
_MARSTAT.xx	Marital Status of Non-Res Member xx	R11799.–R11821.
_RACE.xx	Race of Non-Res Member xx	R11845.–R11865.
_RELATION.xx	Relationship of Non-Res Member xx to Youth	R11866.–R11883.
_SEX.xx	Gender of Non-Res Member xx	R11884.–R11906.
_UID.xx	Unique ID of Non-Res Member xx	R11907.–R11929.

BIOCHILD Roster (Youth's children)
PREFIX = "BIOCHILD" (e.g., BIOCHILD_BDATE.01)

Variable Name	Title: All end in (Ros Item)	Reference #
_BDATE.xx	Birthdate of R Bio Child xx, Final	R05202.01–R05203.02
_DEAD.xx	Is R Bio Child xx Deceased, Final	R05204.–R05205.
_RESIDE.xx	Does R Bio Child xx Reside in Household, Final	R05212.–R05213.
_SEX.xx	Gender of R Bio Child xx, Final	R05214.–R05215.
_ID.xx	ID Number of Biochild xx	R05216.–R05217.

YEMP Roster (Youth's employers)
PREFIX = "YEMP" (e.g., YEMP_CURFLAG.01)

Variable Name	Title: All end in (Ros Item)	Reference #
_CURFLAG.xx	Was R Currently Employed at Interview Date Job xx	R05252.–R05258.
_INDCODE.XX	Type of Business or Industry xx	R13050.–R13056.
_INTERN.xx	Is This an Internship Employer Job xx	R05265.–R05271.
_OCCODE.XX	Rs Occupation/Job Title xx	R13057.–R13063.
_STARTDATE.xx	Employer Start Month/Day/Year Job xx	R05297.–R05303.02
_STOPDATE.xx	Employer Stop Month/Day/Year Job xx	R05304.–R05310.02
_UID.xx	Employer Unique ID Number Job xx	R05311.–R05317.

NOTE: No MILCODE, MILFLAG, or MILPAY variables appear on the round 1 YEMP roster because no respondents reported a military job in round 1. These variables first appear in round 2.

PLACES Roster (Places not enrolled/employed youth goes)
PREFIX = "PLACES" (e.g., PLACES_GETHOME.01)

Variable Name	Title: All begin with PLACES and end in (Ros Item)	Reference #
_GETHOME.xx	Time R Arrives Home after Regular Daily Activity xx	R05228.–R05231.
_GETTIME.xx	Time R Leaves Home for Regular Daily Activity xx	R05232.–R05235.

FREELANCE Roster (Youth's freelance jobs)

Variable Name	Title	Reference #
FREELANCE_JOBS-COD.xx	FREELANCE, Job Code Frame xx (Ros Item)	R13029.–R13036.

YOUTH Roster (Youth respondent information)

PREFIX = "YOUTH" (e.g., YOUTH_ADOPDADID.01)

Variable Name	Title: All end in (Ros Item)	Reference #
_ADOPDADID.01	R 01 Adoptive Dads ID	R05318.
_ADOPMOMID.01	R 01 Adoptive Moms ID	R05319.
_BOTHBIO.01	Does R 01 Live with Both Bio Parents?	R05322.
_DADID.10	R 01 Bio Dads ID	R05323.
_EMANCIPAT.01	Is R 01 Emancipated?	R05326.
_FOSTDADID.01	R 01 Foster Dads ID	R05327.
_FOSTMOMID.01	R 01 Foster Moms ID	R05328.
_GRADE.01	R 01 Current Grade	R05329.
_HHADOPTKID.01	Does R 01 Have Any Adopted Kids in HH?	R05330.
_HHBIOKID.01	Does R 01 Have Any Bio Kids in HH?	R05331.
_HHID.01	R 01 HH Id Number	R05332.
_HHSTEPKID.01	Does R 01 Have Any Step Kids?	R05333.
_ID.01	R 01 ID Number	R05334.
_MOMID.01	R 01 Bio Moms ID	R05336.
_NONR1DEAD.01	Is R 01 1st Non-Resp Bio Parent Deceased?	R05338.
_NONR1ID.01	ID of R 01 1st Non-Resp Bio Parent	R05339.
_NONR1INHH.01	Does 1st Non-Resp Bio Parent of R 01 Live in HH?	R05340.
_NONR1SEX.01	Gender of R 01 1st Non-Resp Bio Parent	R05342.
_NONR2DEAD.01	Is R 01 2nd Non-Resp Bio Parent Deceased?	R05343.
_NONR2ID.01	R 01 2nd Non-Resp Bio Parents ID	R05344.
_NONR2INHH.01	Is R 01 2nd Non-Resp Parent in HH	R05345.
_NONR2SEX.01	Gender of R 01 2nd Non-Resp Bio Parent	R05347.
_NRADOPTKID.01	Does R 01 Have Any Non-Resident Adopted Kids	R05348.
_NRBIOKID.01	Does R 01 Have Any Non-Resident Bio Kids?	R05349.
_NRDADID.01	ID of R 01 Non-Resident Bio Dad	R05350.
_NRMOMID.01	ID of R 01 Non-Resident Bio Mom	R05351.
_NRSTEPKID.01	Does R 01 Have Any Non-Resident Step Kids	R05352.
_PARENT.01	Relationship of Resp Parent to R 01	R05353.
_PARENTGUAR.01	Does R 01 Have a Resp Parent or Guardian in HH	R05354.
_PARENTID.01	ID of R 01 Resp Parent	R05355.
_PARENTSEX.01	Gender of R 01 Resp Parent	R05356.
_SPOPARID.01	ID of R 01 Spouse or Partner	R05358.
_STEPDADID.01	ID of R 01 Step Dad	R05359.
_STEPMOMID.01	ID of R 01 Step Mom	R05360.

PARHHI Roster (Household information in parent interview)
PREFIX = "PARHHI" (e.g., PARHHI_AGE.01)

NOTE: Many of the variables on this roster also appear on the HHI2 roster. The HHI2 roster has been cleaned, while this roster has not. Researchers may wish to use the HHI2 roster rather than the PARHHI roster in analyses.

Variable Name	Title: All end in (Par Ros Item)	Reference #
_AGE.xx	Age of HH Member xx as of Interview Date	R06945.–R06953.
_AGEDOL.xx	Age of HH Member xx as of 12/31/1996	R06954.–R06962.
_DADID.xx	Member xx Bio Dads ID	R06963.–R06971.
_DOB.xx	Member xx Date of Birth	R06972.01–R06980.02
_DOLEL.xx	Is HH Member xx DOL Eligible (Preliminary)	R06981.–R06989
_ELIGIBLE.xx	Member xx DOL, ETP or STP Eligible	R06990.–R06998.
_EMPLOYED.xx	Employment Status of HH Member xx	R06999.–R07001.
_ENROLLSTAT.xx	Is HH Member xx Currently Enrolled	R07002.–R07009.
_ETPEL.xx	Is HH Member xx ETP Eligible (Preliminary)	R07010.–R07018.
_GRADE.xx	Member xx Current Grade	R07019.–R07027.
_HIGHGRADE.xx	Member xx Highest Grade Completed	R07028.–R07036.
_ID.xx	ID of HH Member xx	R07037.–R07045.
_MARSTAT.xx	Member xx Marital Status	R07046.–R07054.
_MOB.xx	Member xx Month of Birth	R07055.–R07063.
_MOMID.xx	Member xx Bio Moms ID	R07064.–R07072.
_PARTNER.xx	Member xx Have a Partner?	R07082.–R07090.
_RACE.xx	Race of HH Member xx	R07091.–R07099.
_RELx.xx, _RELxx.xx	Relationship of Person xx to HH Member xx The relationship variables are provided for people 1-20. For example, REL1.xx provides the relationship of person 1 to household member xx. REL20.xx provides the relationship of person 20 to household member xx.	R07100.–R07196.
_REVDOLEL.xx	Is HH Member xx DOL Eligible (Revised)	R07260.–R07268.
_REVETPEL.xx	Is HH Member xx ETP Eligible (Revised)	R07269.–R07277.
_REVSTPEL.xx	Is HH Member xx STP Eligible (Revised)	R07278.–R07286.
_SEX.xx	Member xx Sex	R07287.–R07295.
_SPOPARID.xx	ID of HH Member xx Spouse or Partner	R07296.–R07304.
_STPEL.xx	Is Member xx STP Eligible (Preliminary)	R07305.–R07313.

PARYOUTH Roster (Youth information for parent interview)
PREFIX = "PARYOUTH" (e.g., PARYOUTH_AGE.01)

Variable Name	Title: All end in (Par Ros Item)	Reference #
_ADOPDADID	Rs Adoptive Dads ID	R07314.
_ADOPMOMID	Rs Adoptive Moms ID	R07315.
_AGE	Age of R as of Interview Date	R07316.
_AGEDOL	Age of R as of 12/31/96	R07317.
_BOTHBIO	Does R Live with Both Bio Parents?	R07318.
_DADID	Rs Bio Dads ID	R07319.
_DOB	Date of Rs Birth	R07320.01–R07320.02
_ELIGIBLE	Is R Eligible for DOL, ETP or STP?	R07321.
_EMANCIPAT	Is R Emancipated?	R07322.
_FOSTDADID	Rs Foster Dads ID	R07323.
_FOSTMOMID	Rs Foster Moms ID	R07324.
_GRADE	Rs Current Grade	R07325.
_HHADOPTKID	Does R Have Any Adopted Kids in HH?	R07326.
_HHBIOKID	Does R Have Any Bio Kids in HH?	R07327.
_HHID	Rs HH ID Number	R07328.
_HHSTEPKID	Does R Have Any Step Kids?	R07329.
_ID	Rs ID Number	R07330.
_MARSTAT	Rs Marital Status	R07331.
_MOMID	Rs Bio Moms ID	R07332.
_NONR1DEAD	Is Rs 1st Non-Resp Bio Parent Deceased?	R07333.
_NONR1ID	ID of Rs 1st Non-Resp Bio Parent	R07334.
_NONR1INHH	1st Non-Resp Bio Parent of R Live in HH?	R07335.
_NONR1SEX	Gender of Rs 1st Non-Resp Bio Parent	R07337.
_NONR2DEAD	Is Rs 2nd Non-Resp Bio Parent Deceased?	R07338.
_NONR2ID	Rs 2nd Non-Resp Bio Parents ID	R07339.
_NONR2INHH	Is Rs 2nd Non-Resp Bio Parent In HH	R07340.
_NONR2SEX	Gender of Rs 2nd Non-Resp Bio Parent	R07342.
_NRADOPTKID	Does R Have Any Non-Resident Adopted Kids	R07343.
_NRBIOKID	Does R Have Any Non-Resident Bio Kids?	R07344.
_NRDADID	ID of Rs Non-Resident Bio Dad	R07345.
_NRMOMID	ID of Rs Non-Resident Bio Mom	R07346.
_NRSTEPKID	Does R Have Any Non-Resident Step Kids	R07347.
_PARENT	Relationship of Resp Parent to R	R07348.
_PARENTGUAR	R Have a Resp Parent or Guardian in HH	R07349.
_PARENTID	ID of Rs Resp Parent	R07350.
_PARENTSEX	Gender of Rs Resp Parent	R07351.
_SEX	Gender of R	R07352.
_SPOPARID	ID of Rs Spouse or Partner	R07353.
_STEPDADID	ID of Rs Stepdad	R07354.
_STEPMOMID	ID of Rs Stepmom	R07355.

ROUND 2

HHI Roster (Household enumeration)
PREFIX = "HHI" (e.g., HHI_AGE.01)

Variable Name	Title: All begin with HHI and end in (Ros Item)	Reference #
_AGE.xx	Age of Household Member xx as of Intdate	R23999.–R24012.
_DEGREE.xx	HH Member xx Highest Degree Earned	R24013.–R24026.
_EMPLOYED.xx	HH Member xx Employment Status	R24032.–R24045.
_ETHNICITY.xx	HH Member xx Ethnicity	R24051.–R24064.
_HHFLAG.xx	HH Member xx Status	R24065.–R24078.
_HIGHGRADE.xx	HH Member xx Highest Grade Completed	R24079.–R24092.
_UID.xx	HH Member xx Unique ID	R24093.–R24106.
_MARSTAT.xx	HH Member xx Marital Status	R24121.–R24134.
_RACE.xx	HH Member xx Race	R24149.–R24162.
_RELY.xx	HH Member xx Relationship to R (Numeric)	R24163.–R24176.
_SEX.xx	HH Member xx Gender	R24191.–R24204.

NONHHI Roster (Non-resident relative enumeration)
PREFIX = "NONHHI" (e.g., NONHHI_AGE.01)

Variable Name	Title: All begin with NONHHI and end in (Ros Item)	Reference #
_AGE.xx	Age of Nonres Member xx as of Intdate	R24205.–R24230.
_ETHNICITY.xx	Is Nonres Member xx of Hispanic Origin?	R24257.–R24277.
_UID.xx	Nonres Member xx Unique ID	R24304.–R24329.
_HHFLAG.xx	Nonres Member xx Status	R24330.–R24355.
_MARSTAT.xx	Nonres Member xx Marital Status	R24357.–R24382.
_RACE.xx	Nonres Member xx Race	R24409.–R24429.
_RELY.xx	Nonres Member xx Relationship to R (Numeric)	R24430.–R24446.
_SEX.xx	Nonres Member xx Gender	R24473.–R24494.

NEWSCHOOL Roster (Respondent's schools)
PREFIX = "NEWSCHOOL" (e.g., NEWSCHOOL_PERIODS.01)

Variable Name	Title: All begin with NEWSCHOOL and end in (Ros Item)	Reference #
_PERIODS.xx	Number of Times R Enrolled in School xx	R24605.–R24610.
_START1.xx	Month/Year R Start 1st Enrollment in School xx	R24611.00–R24616.01
_START2.xx	Month/Year R Start 2nd Enrollment in School xx	R24617.00–R24620.01
_START3.xx	Month/Year R Start 3rd Enrollment in School xx	R24621.00–R24621.01
_STOP1.xx	Month/Year R End 1st Enrollment in School xx	R24622.00–R24627.01
_STOP2.xx	Month/Year R End 2nd Enrollment in School xx	R24628.00–R24631.01
_STOP3.xx	Month/Year R End 3rd Enrollment in School xx	R24632.00–R24632.01
_SCHCODE.xx	School Code Elementary, Middle, High, College	R24633.–R24638.
_INTERVIEW.xx	Which Survey Round School xx Reported in	R24639.–R24644.
_TYPE.xx	Type of School xx R has Attended	R24645.–R24650.
_PUBID.xx	PUBID of School xx R has Attended	R24651.–R24656.
_LEFT.xx	Reason Left School xx	R27260.–R27265.

YEMP Roster (Respondent's employers)
PREFIX = "YEMP" (e.g., YEMP_CURFLAG.01)

Variable Name	Title: All begin with YEMP and end in (Ros Item)	Reference #
_CURFLAG.xx	Is R Currently Working for Employer xx	R24681.–R24689.
_INDCODE.xx	Type of Business or Industry xx	R27266.–R27274.
_INTERN.xx	Employer xx is an Internship	R24752.–R24760.
_MILFLAG.xx	Is Employer xx a Military Employer	R24690.–R24698.
_MILCODE.xx	Employer xx Military Code	R24699.–R24706.
_MILPAY.xx	Employer xx Military Pay Grade	R27236.–R27244.
_OCCODE.xx	Rs Occupation/Job Title xx	R27275.–R27283.
_STARTDATE.xx	Month/Year Start Working for Employer xx	R24725.00–R24733.02
_STOPDATE.xx	Month/Year Stop Working for Employer xx	R24734.00–R24742.02
_UID.xx	Employer xx Unique ID	R24761.–R24769.

FREELANCE Roster (Respondent's freelance jobs)
PREFIX = "FREELANCE" (e.g., FREELANCE_JOBS-COD.01)

Variable Name	Title: All begin with FREELANCE and end in (Ros Item)	Reference #
_JOBS-COD.xx	Job Code Frame xx	R24776.–R24781.
_STARTDATEL.xx	Start Date Reported at DLI Job xx	R24782.00–R24782.01
_STARTDATEC.xx	Start Date Current Freelance Job xx	R24783.00–R24787.01
_STOPDATECU.xx	Stop Date Current Freelance Job xx	R24788.00–R24793.01
_CURATLI.xx	Job was Current at DLI Job xx	R24794.–R24798.
_CURRNOW.xx	Job xx is Current Freelance Job	R24799.–R24804.
_STARTDATEI.xx	Start Date Info Job xx	R24805.–R24809.

BIOCHILD Roster (Respondent's children)
PREFIX = "BIOCHILD" (e.g., BIOCHILD_SEX.01)

Variable Name	Title: All begin with BIOCHILD and end in (Ros Item)	Reference #
_SEX.xx	Bio Child xx Sex	R24906.–R24908.
_BDATE.xx	Bio Child xx Bdate	R24909.01–R24911.02
_DEAD.xx	Bio Child xx is Deceased	R24912.–R24914.
_RESIDE.xx	Bio Child xx Resides in Rs Household	R24915.–R24917.
_ID.xx	Bio Child xx ID	R24918.–R24920.
_UID.xx	Bio Child Unique Household ID Number xx ID	R24921.–R24923.

Additional Roster Variables

Variable Name	Title: All end in (Ros Item)	Reference #
OLDR1NONHHIPARENT _UID	UID of DLI Parent xx This variable permits identification of parents in the parent loops at the beginning of the HHI section. See the note in the codebook for more information.	R27212.-R27216.
OLDR1PARENT	UID of DLI Parent xx This variable has the same function as the above variable.	R27229.-R27232.

ROUND 3

HHI Roster (Household enumeration)
PREFIX = "HHI" (e.g., HHI_AGE.01)

Variable Name	Title: All begin with HHI and end in (Ros Item)	Reference #
_AGE.xx	Age of Household Member xx as of Intdate	R37082.–R37095.
_CHILDID.xx	CHILDID HH Member xx	R37107.–R37117.
_DEGREE.xx	HH Member xx Highest Degree Earned	R37118.–R37131.
_EMPLOYED.xx	HH Member xx Employment Status	R39513.–R39525.
_ENROLLSTAT.xx	Is HH Member xx Currently Enrolled	R37132.–R37145.
_ETHNICITY.xx	HH Member xx Ethnicity	R37146.–R37159.
_HHFLAG.xx	HH Member xx Status	R37160.–R37173.
_HIGHGRADE.xx	HH Member xx Highest Grade Completed	R37174.–R37187.
_UID.xx	HH Member xx Unique ID	R37199.–R37212.
_ID.xx	HH Member xx ID	R37213.–R37226.
_MARSTAT.xx	HH Member xx Marital Status	R37227.–R37240.
_PARENTUID.xx	HH Member xx Parent UID	R37188.–R37198.
_RACE.xx	HH Member xx Race	R37255.–R37268.
_RELY.xx	HH Member xx Relationship to R (Numeric)	R37269.–R37282.
_SEX.xx	HH Member xx Gender	R37297.–R37310.

NONHHI Roster (Non-resident relative enumeration)
PREFIX = "NONHHI" (e.g., NONHHI_AGE.01)

Variable Name	Title: All begin with NONHHI and end in (Ros Item)	Reference #
_AGE.xx	Age of Nonres Member xx as of Intdate	R37311.–R37332.
_CHILDID.xx	Nonres Member xx Child ID	R37348.–R37362.
_ETHNICITY.xx	Is Nonres Member xx of Hispanic Origin?	R37382.–R37406.
_UID.xx	Nonres Member xx Unique ID	R37431.–R37456.
_HHFLAG.xx	Nonres Member xx Status	R37457.–R37482.
_LIVEDWITH.xx	Nonres Member xx Live with R since DLI	R37483.–R37500.
_MARSTAT.xx	Nonres Member xx Marital Status	R37501.–R37524.
_RACE.xx	Nonres Member xx Race	R37565.–R37589.
_RELY.xx	Nonres Member xx Relationship to R (Numeric)	R37590.–R37613.
_SEX.xx	Nonres Member xx Gender	R37640.–R37663.

NEWSCHOOL Roster (Respondent's schools)
PREFIX = "NEWSCHOOL" (e.g., NEWSCHOOL_PERIODS.01)

Variable Name	Title: All begin with NEWSCHOOL and end in (Ros Item)	Reference #
_PERIODS.xx	Number of Times R Enrolled in New School	R37734.–R37740.
_START1.xx	Month/Year R Start 1st Enrollment in School xx	R37741.00–R37747.01
_START2.xx	Month/Year R Start 2nd Enrollment in School xx	R37748.00–R37751.01
_START3.xx	Month/Year R Start 3rd Enrollment in School xx	R37752.00–R37753.01
_STOP1.xx	Month/Year R End 1st Enrollment in School xx	R37754.00–R37760.01
_STOP2.xx	Month/Year R End 2nd Enrollment in School xx	R37761.00–R37764.01
_STOP3.xx	Month/Year R End 3rd Enrollment in School xx	R37765.00–R37766.01
_SCHCODE.xx	School Code xx Elementary, Middle, High, College	R37767.–R37773.
_INTERVIEW.xx	Which Survey Round School xx Reported in	R37774.–R37780.
_TYPE.xx	Type of School xx R has Attended	R37781.–R37787.
_PUBID.xx	PUBID of School xx R has Attended	R37788.–R37794.
_LEFT.xx	Reason Left School xx	R39549.–R39555.

YEMP Roster (Respondent's employers)
PREFIX = "YEMP" (e.g., YEMP_CURFLAG.01)

Variable Name	Title: All begin with YEMP and end in (Ros Item)	Reference #
_CURFLAG.xx	Is R Currently Working for Employer xx	R37819.–R37827.
_INDCODE.xx	Type of Business or Industry xx	R39556.–R39564.
_INTERN.xx	Employer xx is an Internship	R37891.–R37899.
_MILFLAG.xx	Is Employer xx a Military Employer	R37828.–R37836.
_MILCODE.xx	Employer xx Military Code	R37837.–R37845.
_MILPAY.xx	Employer xx Military Pay Grade	R39258.–R39266.
_OCCODE.xx	Rs Occupation/Job Title xx	R39565.–R39573.
_STARTDATE.xx	Month/Year Start Working for Employer xx	R37864.00–R37872.02
_STOPDATE.xx	Month/Year Stop Working for Employer xx	R37873.00–R37881.02
_UID.xx	Employer xx Unique ID	R37900.–R37908.

FREELANCE Roster (Respondent's freelance jobs)
PREFIX = "FREELANCE" (e.g., FREELANCE_JOBS-COD.01)

Variable Name	Title: All begin with FREELANCE and end in (Ros Item)	Reference #
_JOBS-COD.xx	Job Code Frame xx	R37915.–R37920.
_STARTDATEC.xx	Start Date Current Freelance Job xx	R37921.–R37926.
_STOPDATECU.xx	Stop Date Current Freelance Job xx	R37927.–R37932.
_CURRNOW.xx	Job xx is Current Freelance Job	R37938.–R37943.
_UID.xx	Freelance Job xx Unique ID	R37949.–R37954.

TRAINING Roster (Respondent's training programs)
PREFIX = "TRAINING" (e.g., TRAINING_UID.01)

Variable Name	Title: All begin with TRAINING and end in (Ros Item)	Reference #
_UID.xx	UID of Training Program xx	R38261.–R38264.
_STARTDATE.xx	Startdate of Training Program xx	R38265.00–R38268.02
_STOPDATE.xx	Stopdate of Training Program xx	R38269.00–R38272.02
_CURFLAG.xx	Training Program xx Current	R38273.–R38276.

BIOCHILD Roster (Respondent's children)
PREFIX = "BIOCHILD" (e.g., BIOCHILD_SEX.01)

Variable Name	Title: All begin with BIOCHILD and end in (Ros Item)	Reference #
_SEX.xx	Bio Child xx Sex	R38039.–R38041.
_BDATE.xx	Bio Child xx Bdate	R38042.01–R38044.02
_DEAD.xx	Bio Child xx is Deceased	R38045.–R38047.
_PARENTUID.xx	Bio Child xx Parent ID	R38051.–R38053.
_RESIDE.xx	Bio Child xx Resides in Rs Household	R38054.–R38056.
_ID.xx	Bio Child xx ID	R38057.–R38059.
_UID.xx	Bio Child Unique Household ID Number xx ID	R38060.–R38062.

Additional Roster Variables

Variable Name	Title: All end in (Ros Item)	Reference #
PREVPARENT_UID	Prev HH Parent Unique Household ID Number xx ID This variable permits identification of parents in the parent loops at the beginning of the HHI section. See the note in the codebook for more information.	R38193.-R38196.
PREVNONHHIPARENT _UID	Prev NONHHI Parent Unique ID Number xx ID This variable has the same function as the above variable.	R38197.-R38204.

ROUND 4

HHI Roster (Household enumeration)
PREFIX = "HHI" (e.g., HHI_AGE.01)

Variable Name	Title: All begin with HHI and end in (Ros Item)	Reference #
_AGE.xx	Age of Household Member xx as of Intdate	R51697.–R51710.
_BDATE.xx	HH Member xx Birthdate	R51711.01–R51720.02
_CHILDID.xx	CHILDID HH Member xx	R51721.–R51730.
_DEGREE.xx	HH Member xx Highest Degree Earned	R51738.–R51749.
_EMPLOYED.xx	HH Member xx Employment Status	R51750.–R51761.
_ENROLLSTAT.xx	Is HH Member xx Currently Enrolled	R51762.–R51775.
_ETHNICITY.xx	HH Member xx Ethnicity	R51776.–R51789.
_FIRSTLIVE.xx	HH Member xx Date First Lived in HH	R51790.00–R51797.01
_HHFLAG.xx	HH Member xx Status	R51798.–R51811.
_HIGHGRADE.xx	HH Member xx Highest Grade Completed	R51812.–R51825.
_PARENTUID.xx	HH Member xx Parent UID	R51826.–R51835.
_RELIGION.xx	HH Member xx Religious Preference	R51836.–R51849.
_UID.xx	HH Member xx Unique ID	R51850.–R51863.
_MARSTAT.xx	HH Member xx Marital Status	R51878.–R51889.
_RACE.xx	HH Member xx Race	R51904.–R51917.
_RELY.xx	HH Member xx Relationship to R (Numeric)	R51918.–R51931.
_SEX.xx	HH Member xx Gender	R51946.–R51959.

NONHHI Roster (Non-resident relative enumeration)
PREFIX = "NONHHI" (e.g., NONHHI_AGE.01)

Variable Name	Title: All begin with NONHHI and end in (Ros Item)	Reference #
_AGE.xx	Age of Nonres Member xx as of Intdate	R52509.–R52534.
_BDATE.xx	HH Member xx Birthdate	R52535.01–52552.02
_CHILDID.xx	Nonres Member xx Child ID	R52533.–R52570.
_ETHNICITY.xx	Nonres Member xx of Hispanic Origin?	R52612.–R52637.
_UID.xx	Nonres Member xx Unique ID	R52664.–R52689.
_HHFLAG.xx	Nonres Member xx Household Status	R52690.–R52715.
_MARSTAT.xx	Nonres Member xx Marital Status	R52737.–R52759.
_PARENTUID.xx	Nonres Member xx Parent UID	R52786.–R52803.
_RACE.xx	Nonres Member xx Race	R52804.–R52829.
_RELY.xx	Nonres Member xx Relationship to R (Numeric)	R52830.–R52855.
_SEX.xx	Nonres Member xx Gender	R52856.–R52881.

NEWSCHOOL Roster (Respondent's schools)
PREFIX = "NEWSCHOOL" (e.g., NEWSCHOOL_PERIODS.01)

Variable Name	Title: All begin with NEWSCHOOL and end in (Ros Item)	Reference #
_LEFT.xx	Reason Left School xx	R52958.–R52962.
_PERIODS.xx	Number of Times R Enrolled in New School	R52968.–R52972.
_START1.xx	Month/Year R Start 1st Enrollment in School xx	R52973.00–R52977.01
_START2.xx	Month/Year R Start 2nd Enrollment in School xx	R52978.00–R52980.01
_STOP1.xx	Month/Year R End 1st Enrollment in School xx	R52986.00–R52990.01
_STOP2.xx	Month/Year R End 2nd Enrollment in School xx	R52991.00–R52993.01
_SCHCODE.xx	School Code xx Elementary, Middle, High, College	R52999.–R53003.
_INTERVIEW.xx	Which Survey Round School xx Reported in	R53004.–R53008.
_TYPE.xx	Type of School xx R has Attended	R53009.–R53013.
_PUBID.xx	PUBID of School xx R has Attended	R53014.–R53018.
_GED.xx	GED Status of College Enrollment in School xx	R53019.–R53022.

YEMP Roster (Respondent's employers)
PREFIX = "YEMP" (e.g., YEMP_CURFLAG.01)

Variable Name	Title: All begin with YEMP and end in (Ros Item)	Reference #
_CURFLAG.xx	Is R Currently Working for Employer xx	R53043.–R53051.
_INDCODE.xx	Type of Business or Industry xx	R53146.–R53154.
_INTERN.xx	Employer xx is an Internship	R53128.–R53136.
_MILFLAG.xx	Is Employer xx a Military Employer	R53052.–R53060.
_MILCODE.xx	Employer xx Military Code	R53061.–R53069.
_MILPAY.xx	Employer xx Military Pay Grade	R53070.–R53073.
_OCCODE.xx	Rs Occupation/Job Title xx	R53155.–R53163.
_SELFEMP.xx	Is Employer xx a Self-Employed Job?	R53092.–R53100.
_STARTDATE.xx	Month/Day/Year Start Working for Employer xx	R53101.00–R53109.02
_STOPDATE.xx	Month/Day/Year Stop Working for Employer xx	R53110.00–R53118.02
_UID.xx	Employer xx Unique ID	R53137.–R53145.

FREELANCE Roster (Respondent's freelance jobs)
PREFIX = "FREELANCE" (e.g., FREELANCE_JOBS-COD.01)

Variable Name	Title: All begin with FREELANCE and end in (Ros Item)	Reference #
_JOBS-COD.xx	Job Code Frame xx	R53169.–R53173.
_STARTDATEC.xx	Start Date Current Freelance Job xx	R53174.00–R53178.01
_STOPDATECU.xx	Stop Date Current Freelance Job xx	R53179.00–R53183.01
_CURRNOW.xx	Job xx is Current Freelance Job	R53188.–R53192.
_UID.xx	Freelance Job xx Unique ID	R53193.–R53197.

TRAINING Roster (Respondent's training programs)
PREFIX = "TRAINING" (e.g., TRAINING_UID.01)

Variable Name	Title: All begin with TRAINING and end in (Ros Item)	Reference #
_UID.xx	UID of Training Program xx	R54514.–R54518.
_STARTDATE.xx	Startdate of Training Program xx	R54519.00–R54523.02
_STOPDATE.xx	Stopdate of Training Program xx	R54524.00–R54528.02
_CURFLAG.xx	Training Program xx Current	R54529.–R54533.

BIOCHILD Roster (Respondent's children)
PREFIX = "BIOCHILD" (e.g., BIOCHILD_SEX.01)

Variable Name	Title: All begin with BIOCHILD and end in (Ros Item)	Reference #
_SEX.xx	Bio Child xx Sex	R53221.–R53223.
_BDATE.xx	Bio Child xx Bdate	R53224.01–R53226.02
-DOD.xx	Bio Child xx Date of Death	R53227.00–53229.01
_DEAD.xx	Bio Child is Deceased	R53230.–R53232.
_PARENTUID.xx	Bio Child xx Parent ID	R53236.–R53238.
_RESIDE.xx	Bio Child xx Resides in Rs Household	R53239.–R53241.
_ID.xx	Bio Child xx ID	R53242.–R53244.
_UID.xx	Bio Child Unique Household ID Number xx ID	R53245.–R53247.

NLSY97 Appendix 9: Family Process and Adolescent Outcome Measures

This appendix, which describes a number of variables created by Child Trends. Inc., is provided in a separate printed document. Interested researchers should contact NLS User Services for more information. The variables discussed in this appendix are as follows:

Question Name	Variable Title
Round 1	
FP_YMFRELAT	Residential Mom Supportive of Residential Dad, Youth Report
FP_YFMRELAT	Residential Dad Supportive of Residential Mom, Yth Report
FP_PPRELAT	Residential Parent Supports Spouse/Partner, Parent Report
FP_YMSUPP	Youth Relationship with Residential Mom, Youth Report
FP_YFSUPP	Youth Relationship with Residential Dad, Youth Report
FP_YNRMSUPP	Youth Relationship with Non-Residential Biological Mom, Youth Report
FP_YNRFSUPP	Youth Relationship with Non-Residential Biological Dad, Youth Report
FP_YHROUTIN	Index of Family Routines, Youth Report
FP_YMMONIT	Degree of Parental Monitoring by Residential Mother, Youth Report
FP_YFMONIT	Degree of Parental Monitoring by Residential Father, Youth Report
FP_YNRMMONIT	Degree of Parental Monitoring by Non-Residential Mother, Youth Report
FP_YNRFMONIT	Degree of Parental Monitoring by Non-Residential Father, Youth Report
FP_YHLIMITS	Autonomy, Control and Limit Setting, Youth Report
FP_PHLIMITS	Autonomy, Control, and Limit Setting, Parent Report
FP_YHBROKED	Limit-Breaking, Youth Report
FP_PHBROKED	Limit-Breaking, Parent Report
FP_YMPSTYL	Residential Mother's Parenting Style, Youth Report
FP_YFPSTYL	Residential Father's Parenting Style, Youth Report
FP_YNRMPSTYL	Non-Residential Mother's Parenting Style, Youth Report
FP_YNRFPSTYL	Non-Residential Father's Parenting Style, Youth Report
FP_PPRELIG	Parent's Religiosity, Parent Report
FP_ADHRISKI	Family/Home Risk Index
FP_ADPENVRI	Physical Environment Risk Index
FP_ADENRCHI	Enriching Environment Risk Index
FP_YYFBEHS	Girls Behavioral/Emotional Problems Scale, Youth Report
FP_YYMBEHS	Boys Behavioral/Emotional Problems Scale, Youth Report
FP_PYFBEHS	Girls Behavioral/Emotional Problems Scale, Parent Report
FP_PYMBEHS	Boys Behavioral/Emotional Scale, Parent Report
FP_YYCRIMI	Deliquency Score Index
FP_YYSUBSI	Substance Use Index
Round 2	
FP_YMFRELAT	Residental Mom Supportive of Residential Dad, Yth Report
FP_YFMRELAT	Residental Dad Supportive of Residential Mom, Yth Report
FP_YMSUPP	Youth Relationship with Residential Mom, Youth Report
FP_YFSUPP	Youth Relationship with Residential Dad Youth Report

Question Name	Variable Title
FP_YHROUTIN	Index of Family Routines, Youth Report
FP_YMMONIT	Degree of Parental Monitoring by Residential Mother, Youth Report
FP_YFMONIT	Degree of Parental Monitoring by Residential Father, Youth Report
FP_YHLIMITS	Autonomy, Control, and Limit Setting, Youth Report
FP_YHBROKED	Limit-Breaking, Youth Report
FP_YMPSTYL	Residential Mothers Parenting Style, Youth Report
FP_YFPSTYL	Residential Fathers Parenting Style, Youth Report
FP_YNRMPSTYL	Non-Residential Mothers Parenting Style, Youth Report
FP_YNRFPSTYL	Non-Residential Fathers Parenting Style, Youth Report
FP_YYCRIMI	Deliquency Score Index
FP_YYSUBSI	Substance Use Index
FP_YYCRIMI_DLI	Deliquency Score Index (DLI)
FP_YYSUBSI_DLI	Substance Use Index (DLI)
Round 3	
FP_YMFRELAT	Residental Mom Supportive of Residential Dad, Yth Report
FP_YFMRELAT	Residental Dad Supportive of Residential Mom, Yth Report
FP_YMSUPP	Youth Relationship with Residential Mom, Youth Report
FP_YFSUPP	Youth Relationship with Residential Dad Youth Report
FP_YHROUTIN	Index of Family Routines, Youth Report
FP_YMMONIT	Degree of Parental Monitoring by Residential Mother, Youth Report
FP_YFMONIT	Degree of Parental Monitoring by Residential Father, Youth Report
FP_YHLIMITS	Autonomy, Control, and Limit Setting, Youth Report
FP_YHBROKED	Limit-Breaking, Youth Report
FP_YMPSTYL	Residential Mothers Parenting Style, Youth Report
FP_YFPSTYL	Residential Fathers Parenting Style, Youth Report
FP_YNRMPSTYL	Non-Residential Mothers Parenting Style, Youth Report
FP_YNRFPSTYL	Non-Residential Fathers Parenting Style, Youth Report
FP_YYCRIMI_DLI	Deliquency Score Index (DLI)
FP_YYSUBSI_DLI	Substance Use Index (DLI)
Round 4	
FP_YHROUTIN	Index Of Family Routines, Youth Report
FP_YMMONIT	Degree Of Parental Monitoring By Residential Mother, Youth Report
FP_YFMONIT	Degree Of Parental Monitoring By Residential Father, Youth Report
FP_YMPSTYL	Residential Mothers Parenting Style, Youth Report
FP_YFPSTYL	Residential Fathers Parenting Style, Youth Report
FP_YNRMPSTYL	Non-Residential Mothers Parenting Style, Youth Report
FP_YNRFPSTYL	Non-Residential Fathers Parenting Style, Youth Report
FP_YYCRIMI_DLI	Deliquency Score Index (Dli)
FP_YYSUBSI_DLI	Substance Use Index (Dli)

Question Name	Variable Title
FP_YMNTHLTH	Youth Mental Health Scale
FP_SEXEVER	R4: Youth Ever Had Sex
FP_SEXAGE	Age At First Sex
FP_SEXYEAR	Year Of First Sex

NLSY97 Appendix 10:

CAT-ASVAB Scores

ADMINISTRATION AND SCORING OF THE CAT-ASVAB

During round 1 of the NLSY97, most respondents participated in the administration of the computer-adaptive form of the Armed Services Vocational Aptitude Battery (CAT-ASVAB). The Department of Defense (DOD) used the NLSY97 sample as part of a larger effort to establish new norms for the CAT-ASVAB, a military enlistment test battery. A total of 7,127 NLSY97 respondents (or 79.3 percent of the NLSY97 sample) completed this test: 5,452, or 80.8 percent, of the cross-sectional sample and 1,675, or 74.9 percent, of the supplemental sample. In addition, young adults in two separate samples selected during the NLSY97 screening took the CAT-ASVAB during the same time period. Details about the administration of the CAT-ASVAB and the two additional samples are provided in the *NLSY97 User's Guide*.

The CAT-ASVAB consists of ten power and two speeded subtests that measure vocational aptitude in the following areas:

- Arithmetic Reasoning
- Assembling Objects
- Auto Information
- Coding Speed
- Electronics Information
- General Science
- Mathematics Knowledge
- Mechanical Comprehension
- Numerical Operations
- Paragraph Comprehension
- Shop Information
- Word Knowledge

The 10 power subtests are administered using an adaptive testing procedure that matches the difficulty level of the individual test items to the ability levels of the respondents. At the beginning of each test, the procedure selects an item of comparable difficulty for all respondents.[1] After the respondent answers the first item, the program computes a provisional estimate of ability taking into account the respondent's answer to the item. The next item is selected in accord with that provisional estimate. After the administration of each subsequent item, the provisional estimate of ability is updated using a sequential Bayesian procedure (Owen, 1969, 1975) and the updated estimate is used to match the difficulty of the next item to the ability level of the respondent. The process continues until the respondent has answered a fixed number of items (Segall et. al., 1997).[2]

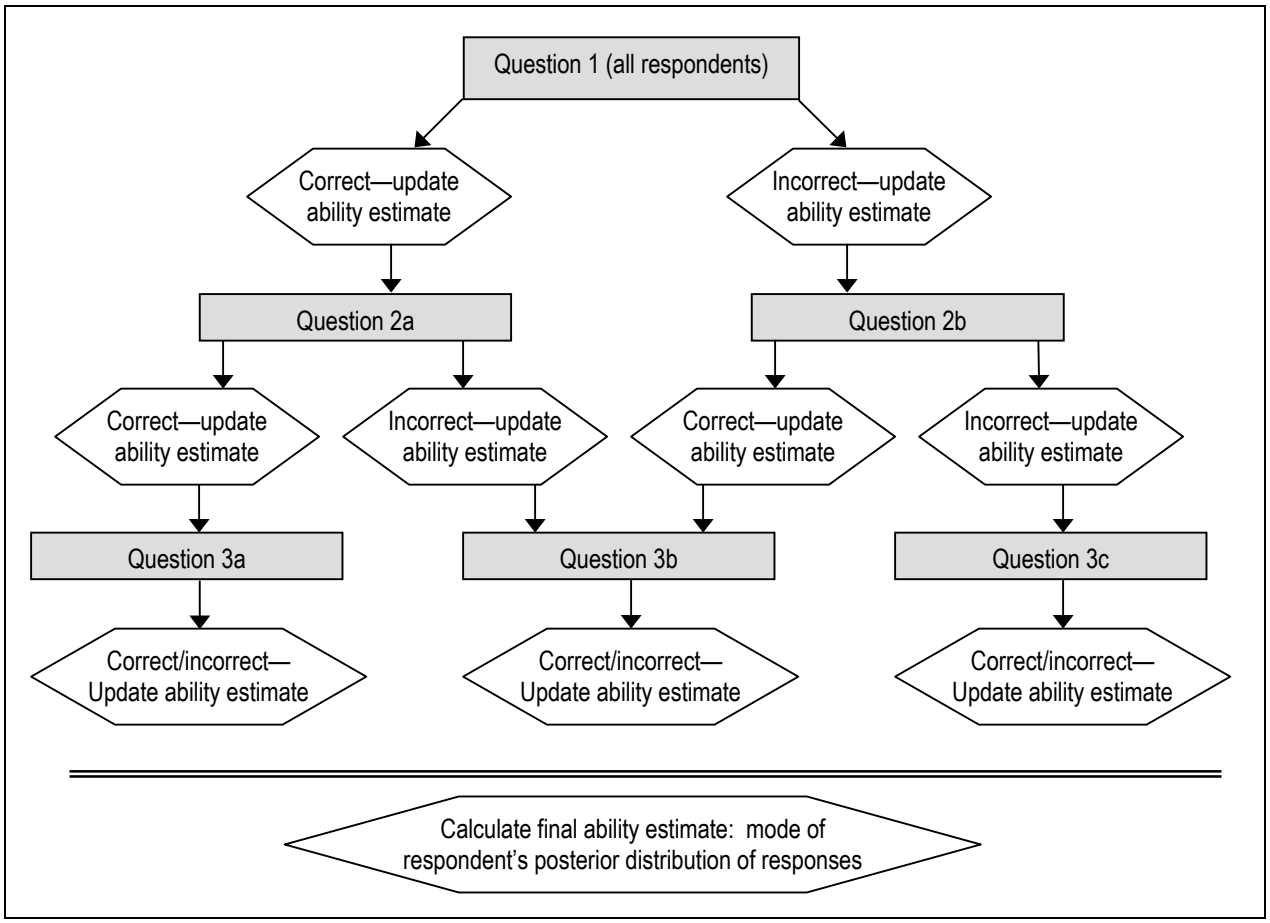
Adaptive testing procedures are desirable because they require fewer items, and thus less testing time than conventional tests to obtain ability estimates of comparable precision. This is because items matched to the ability level of a respondent provide more information about the respondent's ability level.

Figure 1 is an illustration of the way an adaptive procedure might work in a simple test where each respondent answers only three questions. **Please note that this is only an example; the actual administration of the CAT-ASVAB is more complex and is not represented here.**

[1] The youngest NLSY97 respondents, those born in 1983 and 1984, were administered an "easy" form of the test in which the first question had a lower level of difficulty.

[2] In each subtest, a few respondents answered less than the fixed number of questions, presumably because they stopped taking the test or they reached the maximum time allowed for that section. These respondents still have final ability estimates, but these scores may be less precise because they are based on fewer questions.

Figure 1. Three-Question Test Administered Using an Adaptive Testing Procedure



Because respondents do not answer the same set of items, conventional scores such as number or percent correct cannot be used to compare their performance. Instead, the program computes an item response theoretic (IRT) estimate of ability that summarizes each respondent's overall performance on the subtest. These scores are computed on a comparable scale and thus, can be compared across respondents—that is, a lower score indicates poorer performance, and a higher score indicates better performance. The final ability estimates are similar to the provisional ability estimates calculated after each test question is answered. Both calculations use IRT estimates of the item parameters obtained in a previous calibration of the CAT-ASVAB items (Segall, Moreno, and Hetter, 1997). However, the final ability estimate is not simply the last update of the provisional ability estimate. It is possible that two respondents who answered the same set of items, with the same correct or incorrect responses, would have slightly different provisional ability estimates because the items were answered in a different order. The final ability estimate takes this question order effect into account. In statistical terms, the ability scores are maximum a posterior (MAP) or Bayes modal estimates of ability. They are estimates of the mode of the posterior distribution of ability for the respondent, given the respondent's pattern of correct and incorrect responses to the subset of items completed. Final ability estimates are sometimes referred to as theta scores.

The variance of respondents' posterior distributions provides information on the precision of the ability estimates. The square root of the posterior variance, called the posterior standard deviation, is similar in meaning to the standard error of measurement and can be used to construct confidence bands on the individual estimates of ability.

The two speeded tests in the CAT-ASVAB, Coding Speed and Numerical Operations, are administered in a non-adaptive format—all respondents answer the same items in the same order. The final ability estimate for these subtests is a rate score based on the proportion of correct responses corrected for guessing, divided by the mean screen presentation time for the items (Segall et al, 1997). The mean presentation time depends on the response latencies of the respondents—that is, how long it takes them to complete the items. The rate score provides a measure of the speed and accuracy of responding.

Researchers interested in more information about test administration and scoring using Item Response Theory may wish to refer to Lord (1980).

Two additional sets of scores are scheduled for release in a future NLSY97 round. The Department of Defense will create normed scores for each section of the ASVAB. These differ from the final ability estimates in that they take into account respondent characteristics such as age and tell how a respondent performed relative to a national sample of other respondents with similar characteristics. Finally, a composite score derived from select sections of the CAT-ASVAB will provide an Armed Forces Qualifications Test score (AFQT), a general measure of trainability and a primary criterion of eligibility for Armed Forces enlistment.

> ***User Notes:*** The ability estimate scores **cannot** be used in analyses in the same way as normed scores or an AFQT score. Researchers who wish to use these ability estimates should consult an expert in Item Response Theory test administration.

CAT-ASVAB DATA ON THE NLSY97 CD

Included for the first time on the round 3 NLSY97 Event History CD, the ASVAB scores and related variables permit users to compare the ability levels of NLSY97 respondents in relation to one another. All ASVAB variables can be located by looking in the "Aptitude Tests" area of interest or by searching for question names that start with "ASVAB." Users should note that, although the survey year is listed on the CD as 1999 (round 3), the ASVAB was actually administered during the round 1 field period. The 1999 listing refers to the fact that the variables were added to the data set in round 3. There are five groups of variables available.

First, a variable for each subtest indicates the number of items the respondent answered in that section of the test. The number of items completed is one factor in the creation of the final ability estimates for the two speeded subtests.

The final ability estimates for each subtest comprise the second set of ASVAB variables. These estimates may have positive or negative values because the scores are on the scale of the original calibration study, which set the mean of the latent ability distribution to 0 and the standard deviation to 1 in the calibration population of respondents. Because negative codes are reserved for missing data in the NLSY97 data set, one variable cannot contain both positive and negative scores. Therefore, the final ability estimates are reported in two separate variables, one for positive scores and one for negative scores. Each respondent will have a valid value for only one of the two variables. Researchers must combine the variables to examine the scores for the full set of respondents.

User Notes: As described above, the score distribution for the final ability estimates is based on the scale of the original calibration sample, which was set to a mean of 0 and a standard deviation of 1 in that population of respondents. The ability estimates do not have a mean of 0 in the NLSY97 sample of respondents. Overall, the ability levels of the NLSY97 respondents tended to be lower than those of the calibration sample, which was comprised of older respondents. Users will therefore notice a significantly higher number of negative scores than positive scores. However, the final ability estimates can still be used to rank the relative ability of NLSY97 respondents.

The third set of ASVAB variables, the posterior variances, provides estimates of the precision of the ability scores. These variables are available only for the 10 power subtests.

The fourth set of variables on the NLSY97 CD reports respondents' answers to the on-line questionnaire. These questions, mostly asked after the respondent had completed the test, collected information about the respondent's background and the testing conditions. Respondents first answered questions about their school experiences, such as the highest grade they had completed and the highest degree attained, their average grades in their last year of school, the subjects they had taken in school, and the quality of teaching in their high school science and shop classes. With respect to respondents' backgrounds, variables report respondents' ethnicities, whether English was their primary language, whether another language was spoken at home, and whether their parents worked for pay during the respondent's childhood. Finally, questions related to the ASVAB administration included whether the respondent had taken the test before, whether he or she had served in the military, the extent of the respondent's computer use prior to test administration, the main reasons the respondent took the test, and whether the respondent did as well as possible. Respondents were also asked to evaluate the comfort and noise level of the test-taking environment and to state whether the test's instructions were clear.

Finally, the data set includes item latencies (or question timings) for each of the items in the on-line questionnaire. These variables, identified by variable titles beginning with "IL," essentially measure how long the question appeared on the computer screen.

User Notes: Respondents who did not take the ASVAB are assigned a –4, valid skip, in the data. For most NLSY97 variables, a valid skip indicates that the respondent was not supposed to be asked a particular question. However, all respondents were eligible for the ASVAB administration, and a valid skip for these variables means that the respondent chose not to participate.

REFERENCES

Lord, F.M. *Application of item response theory to practical testing problems*. Hillsdale, NJ: Erlbaum, 1980.

Owen, R.J. *A Bayesian approach to tailored testing* (RB-69-92). Princeton, NJ: Educational Testing Service, 1969.

Owen, R.J. "A Bayesian sequential procedure for quantal response in the context of adaptive mental testing." *Journal of the American Statistical Association* 70 (1975): 351–56.

Segall, D.O., Moreno, K.E., Bloxom, B., and Hetter, R.D. Psychometric procedures for administering CAT-ASVAB. In W.A. Sands, B.K. Waters, and J.R. McBride (eds.), *Computerized adaptive testing: From inquiry to operation.* Washington, DC: American Psychological Association, 1997.

Segall, D.O., Moreno, K.E., and Hetter, R.D. Item pool development and evaluation. In W.A. Sands, B.K. Waters, and J.R. McBride (eds.), *Computerized adaptive testing: From inquiry to operation.* Washington, DC: American Psychological Association, 1997.

NLSY97 Appendix 11:

Collection of the Transcript Data, Wave 1

To complement data on respondents' educational experiences collected during the yearly interviews, NLSY97 staff collect transcripts directly from respondents' high schools as the youths graduate or leave school. When the transcripts are received from the schools, survey staff code the information they contain into a standard format. The resulting created variables comprise a history of the respondent's terms in school, courses taken, and other academic indicators. This appendix describes the survey materials used during data collection and explains the procedures and criteria for data entry and coding. It also lists specific details about individual Transcript Survey variables and provides coding information for the course code variables.

TRANSCRIPT SURVEY DATA COLLECTION

Conducted in 1999–2000, wave I of the NLSY97 Transcript Survey sought hard copy transcripts from 1,622 NLSY97 respondents who had provided signed authorization for transcript collection, and who were no longer enrolled in high school in spring 2000. Non-enrollment occurred when the youth either graduated from high school or dropped out of school and was at least 18 years old. In wave I, coded transcript data are available for 1,417 respondents. Wave II will collect additional transcripts in winter 2003–04.

NORC mailed a transcript request packet to each school from which an NLSY97 youth received his or her high school diploma, or to the last school the youth attended. The packet contained informational materials about the NLSY97, a description of the NLSY97 Transcript Survey, and documentation of administrative permission from districts to contact schools. In addition, packets included the following items, reproduced on the following pages:

1. a cover letter addressed to the school principal
2. a one-page cover sheet questionnaire collecting school-specific grading and transcript policies
3. a Student Request list identifying the sampled students in the school
4. the signed permission forms for these students

Figure 1. Cover letter for Transcript Survey materials

February 3, 2000

Dear Principal:

The National Opinion Research Center (NORC) at the University of Chicago is conducting a survey called the National Longitudinal Survey of Youth 1997 (NLSY97). The survey is intended to collect, on a nationwide basis, information pertaining to the employment, job training, education, and family experiences of young adults born from 1980 to 1984. Sponsored by the U.S. Department of Labor and supervised by the Bureau of Labor Statistics, the survey provides researchers with data relevant for examining linkages between family, education, and employment outcomes.

While we collect a great amount of information about the home environment for these youth, it also is vital that we learn about their school environment and academic success. For this reason, the U.S. Department of Labor is asking for your voluntary cooperation in providing information about the school experiences of these youth.

Specifically, we would appreciate your help in providing the following:

- School transcripts for the student(s) listed on the enclosed permission forms
- Answers to the enclosed one-page questionnaire that will provide additional information to help us interpret the transcript information we collect from schools across the nation
- A copy of your course catalog or a listing of your course offerings

Enclosed is a copy of the signed permission form for each student for whom we are requesting information. NORC obtained signatures from the student and from the parents of all students under the age of 18 years. For students 18 years of age and older, we accepted the youth's signature as adequate consent. NORC is sensitive to the need to maintain the confidentiality of all respondents in this survey. Identifying information is protected in accordance with the Privacy Act statement provided on the back of this form. In addition, we have enclosed a brochure to provide additional information about this important longitudinal research effort.

NORC estimates that it will take an average of 30 minutes per student to complete the enclosed questionnaire and prepare a copy of the transcript information.

Please fill out the enclosed reimbursement form, which allows you to be reimbursed in the amount of $10.00 per student for any costs associated with reproducing transcripts. Also enclosed is a postage-paid business reply envelope for returning the student list, transcript(s), completed questionnaire, and course listings to NORC.

We look forward to receiving your completed documents within the next two weeks. Should you have any questions or comments, please call our project information line at 1-877-400-0449. You may call at any time; if we are not able to assist you at the time of your call, we will return your call within one business day. You may also write to the Bureau of Labor Statistics, National Longitudinal Survey, 2 Massachusetts Avenue N.E., Washington, DC 20212. The OMB clearance number for this survey is 1220-0157. Without this current OMB number, BLS could not conduct this survey.

We appreciate your support and thank you in advance for your cooperation.

Sincerely,

Michael Horrigan
Program Director, BLS

A. Rupa Datta
Project Director, NORC

Figure 2. Cover sheet for school-specific information

TRANSCRIPTS COVER SHEET

NORC/4896
1/00

National Longitudinal Survey of Youth 1997 - Transcript Survey Supplement

SCHOOL NAME: EXAMPLE HIGH SCHOOL
SCHOOL ID:

PREPARER'S NAME: _____ DATE PREPARED: _____

PREPARER'S TELEPHONE NUMBER: (_____)_____ E-MAIL ADDRESS: _____

Please answer the following questions about the transcripts being provided. Your answers will help us analyze the transcripts.

1. **Please specify below your school's grade scale (the percentage equivalent of each letter grade).** (If your school uses different grade scales for different difficulty levels of courses or tracks (honors, general, special education, etc.), please specify the scale associated with each. If your school has only one scale, record the information in the first column.)

LEVEL_____	LEVEL_____	LEVEL_____	LEVEL_____
A = _____% to _____%	A = _____% to _____%	A = _____% to _____%	A = _____% to _____%
B = _____% to _____%	B = _____% to _____%	B = _____% to _____%	B = _____% to _____%
C = _____% to _____%	C = _____% to _____%	C = _____% to _____%	C = _____% to _____%
D = _____% to _____%	D = _____% to _____%	D = _____% to _____%	D = _____% to _____%
F = _____% to _____%	F = _____% to _____%	F = _____% to _____%	F = _____% to _____%

2. **Is any of the information below unavailable at your school, and consequently not included on the transcripts enclosed? Check off each variable that is <u>unavailable</u>.** (Please retrieve any information that is available at your school from other sources, such as guidance records, and write the information on each student's transcript.)

3. **<u>Student-level information</u>**

_____ number of absences per year or term
_____ rank in class and class size
_____ date student left school
_____ reason student left school (graduated, transferred, etc.)
_____ GPA
standardized test scores for the:
_____ PSAT _____ ACT _____ SAT _____ AP

<u>Course-taking histories for grades 9 (or 10) through 12, including:</u>

_____ course title
_____ grade level, year, and term course taken
_____ number of credits earned
_____ grade assigned

4. **Please feel free to record below any information about the transcripts enclosed that you believe would be helpful in processing them.**

_____

_____

_____

Thank you for your help. We appreciate your cooperation.

Figure 3. Student Request list identifying sample students

2/03/00

STUDENT TRANSCRIPT REQUEST LIST

National Longitudinal Survey of Youth 1997 - Transcript Survey Supplement

School Name: EXAMPLE HIGH SCHOOL
School ID#: XXXXXXXX
State: ST

INSTRUCTIONS: Please return transcripts for each student listed below. Please indicate whether he/she has participated in a special education, bilingual education, or gifted course or program by circling yes or no. If you are unable to provide a transcript for a student, please indicate the reason (such as "never attended", "transferred") in the "COMMENTS" column to the right of the student's name. If a student transferred to or from another school, please return any transcripts that you have, and write in the name, city and state of the transfer school in the "COMMENTS" column.

PLEASE ENCLOSE THIS DOCUMENT WHEN RETURNING TRANSCRIPTS.

STUDENT INFORMATION	Has the student participated in any courses or programs in the following areas? (PLEASE CIRCLE ANSWER.)			TRANSCRIPT INFORMATION	
Please supply transcripts and answer questions for each student listed below:	Special Education?	Bilingual Education?	Gifted Courses or Program?	Transcript Returned?	COMMENTS: (Additional information? Reason transcript not returned? If applicable, transfer school name, city, state.)
ID#: XXXXXXX1 Student Name: LAST NAME III, FIRST NAME Birth Date: XX/XX/19XX Attendance Last Verified: XX/19XX	Yes No	Yes No	Yes No	Yes No	
ID#: XXXXXXX2 Student Name: LAST NAME III, FIRST NAME Birth Date: XX/XX/19XX Attendance Last Verified: XX/19XX	Yes No	Yes No	Yes No	Yes No	

Figure 4. Student permission form

13

|___|___|___|___|___|___|___|___|
Youth ID

NORC 4893
OMB No: 1220-0157
Expiration Date: 8/31/2001

Student's Name_____
(Please print)

NLSY97 ROUND 4
PERMISSION TO CONTACT SCHOOL

On behalf of the U.S. Department of Labor, the National Opinion Research Center (NORC) requests permission to obtain information from school transcripts and related school records for the student named above. NORC also requests permission to contact the school regarding the general characteristics of the school as well as some specific characteristics of some classes that the student attends/attended.

Parent
NORC has permission to obtain school transcript, related school records, and school/class characteristic information pertaining to my child from my child's school. Identifying information is protected in accordance with the Privacy Act statement provided on the back of this form.

Parent's Signature:_____ Date:_____

Student
NORC has permission to obtain school transcript, related school records, and school/class characteristic information pertaining to me from my school. Identifying information is protected in accordance with the Privacy Act statement provided on the back of this form.

Student's Signature:_____ Date:_____

WHITE COPY - MAIL TO CO WITH CASE MATERIALS / YELLOW COPY - MAIL TO FM / PINK COPY - KEEP FOR YOUR RECORDS

Figure 4 continued. Student permission form (reverse side)

PRIVACY ACT STATEMENT

In accordance with the Privacy Act of 1974, as amended (5 U.S.C.552a), you are hereby notified that this survey is sponsored by the U.S. Department of Labor, Bureau of Labor Statistics, under authority of 29 U.S.C.2. Your participation is vital to the success of the survey and is voluntary. The information collected in this survey will be used by BLS and other researchers to understand the labor market experience of individuals over time. BLS makes available to the general public and other federal agencies conducting labor force research information that are not individually identifiable. Under written agreements to protect the confidentiality and security of identifying information, BLS provides potentially-identifying geographic information to researchers to conduct specific research projects that further the mission and function of BLS. Unless required by law, or necessary for litigation or legal proceedings, and except as provided in the Privacy Act or as indicated in this statement, we will hold personal identifiers (e.g., name, address, or social security number) in confidence and will not release them to the public. BLS may release identifying data to the National Opinion Research Center (NORC) and the Ohio State University, and contractors, grantees, and their employees or volunteers who are working on this project for BLS and who need access to the material, or to the National Archives and Records Administration or the General Services Administration for records management purposes.

CREATION OF THE TRANSCRIPT DATA FILE

Organization of the data. There are several different types of variables in the transcript data file. First, at the school level the variable TRANS_SCH_CAT reports whether a course catalog was received from the school to aid in coding. The highest number of schools reported for any respondent is 12, so this variable is repeated 12 times. This course catalog variable also functions as the identification number of the school. During the coding process described below, each school attended by a respondent was assigned an ID number between 1 and 12, with the school that provided the transcript always listed as school #01. These numbers were used in variables that report which school the respondent attended in each term—for example, if a respondent has a value of 4 for term 1, then he or she attended school #04 in the course catalog variables. This ID number does not link to any variables in the main data file.

Second, the transcript file includes information about the respondents that is not associated with a specific term or course. For example, these variables present test scores on a variety of achievement tests (ACT, PSAT, SAT, SAT II, AP), information on absences and tardies, the student's school completion status, and dates of enrollment. Variables also indicate whether the respondent participated in programs such as gifted, bilingual, or special education.

A number of variables refer to the respondent's terms of enrollment. For up to 18 terms, these items report the beginning and ending dates of the term, the way in which the school year is divided (such as a season, semester, entire year, or another system), the academic year of the term, the respondent's grade level that term, and the number of credits earned. A variable listing the school the respondent attended during that term can be linked to the course catalog variable as described above.

Finally, the transcript file provides details about each course appearing on a student's high school transcript. Course-specific variables include the course code from the Revised Secondary School Taxonomy (SST-R), the grade earned in the course, and the credit value of the course. Because schools use many different grading systems, the course grades were converted into a standard scale that can be compared across respondents. A series of variables called "Recoding Status of Grade" indicates how the grade earned variable for each course was created. This process is described in more detail below.

Data entry and processing procedures. The transcript data capture process involved several distinct data entry steps, tailored to the structure of the data, the cleaning and reconciliation needs for the relevant variables, and scheduling requirements of the data collection process. The basic data entry and processing steps were:

1) Entry of course-level data into an Access data capture system from high school transcripts
2) Coding of entered course-level data using Access coding system
3) Entry of student-level data from Student Request List and high school transcripts into NORC's SurveyCraft Computer-Assisted Data Entry (CADE) system
4) Entry and coding of transfer school information from Student Request List, high school transcripts, and NLSY97 youth interview data using Access and SAS programs
5) Entry of school-level data from one-page Transcript Cover Sheet into SurveyCraft CADE system
6) Assigning course grades to a uniform grade scale using SAS transformations

Course-level data entry. Course-level data include the course title, course number (assigned by school), grade earned, credits earned, and honors designation. For matching purposes, the school ID was assigned and term dates were captured during this phase of data entry. Entry was done into an Access data-capture system. All courses were independently entered twice. Where entry and re-entry matched perfectly, no further quality control was performed. If one or more discrepancies were found electronically between the entry and re-entry, a supervisor adjudicated the two data-entered versions with the original hard copy transcript to determine the accurate values. Courses were entered in the order that they appeared on the transcript. This order varied

from school to school, with systems including chronologically, alphabetically by course title, numerically by course number, etc.

Course coding. Course-level data were used for coding courses into the Revised Secondary School Taxonomy (SST-R), a hierarchical framework for high school course offerings. After all course-level data from a transcript had been entered, re-entered, and adjudicated, the transcript was available for course coding. To maximize coder familiarity with school naming and catalog conventions, all transcripts from a school were usually coded together. Coding of all courses was done independently by two coders. If the two codes were not equal, a supervisor adjudicated the discrepancy and assigned a final code. Because many schools did not submit course catalogs or had indecipherable course titles (e.g., Course 1), coders called some schools directly for assistance in coding, speaking to administrative or instructional staff who were able to clarify course content. The coding process used a menu-driven Access system, which exploited the hierarchical structure of the code frame and prevented coders from inadvertently entering invalid codes. All 'uncodable' courses were reviewed by the coding supervisor and project director where necessary.

Student-specific data down to the term level. All other student-specific data were captured in a SurveyCraft instrument for computer-assisted data entry. These variables include the student's enrollment in gifted, special education, or bilingual programs, standardized test scores, dates of enrollment at the school, class rank and cumulative grade-point average, term-level information on beginning and ending dates of terms, absences and tardies, and credits earned by term. The SurveyCraft program generated a single record for each youth, containing up to 18 terms of study. Term date information was used to match term-level data with the school attended during that term. All transcripts from a school were CADEd at the same time to exploit coder familiarity with transcript formats and school-specific abbreviations. All transcripts were independently entered twice. Where entry and re-entry matched perfectly, no further quality control was performed. If one or more discrepancies were found electronically between the entry and re-entry, a supervisor adjudicated the two data-entered versions with the original hard copy transcript to determine the accurate values. Terms were entered in chronological order when such sequence could be determined.

Transfer data. Transcripts often included information about courses attended at other institutions. These data could appear either as an original hard copy attachment to the sampled school's transcript or as additional lines on the sampled school's transcript. These terms and courses were CADEd during the appropriate stage of data entry, with a designation that the term or course pertained to a transfer school. Course and term-specific information about transferred work was generally complete, but information about the school from which work was transferred was often inadequate for coding purposes. As described above, all terms attended at the same school are associated with the same school ID.

School data. The one-page Transcript Cover Sheet provided information for assigning course grades to a uniform grade scale. These data were entered into a SurveyCraft data capture instrument, once for each school submitting valid transcripts. Ten percent of schools were re-entered, and a supervisor referred to the original hard-copy to adjudicate discrepancies.

Course grades. High school transcripts included a variety of systems for course grades, including letter grades or numbers. For ease of comparison, these were standardized into a uniform grading system. The standardized grading scale for the resulting CRS_GRADE variable ranges from 01 to 20. Table 1 lists the corresponding letter grades for each of the CRS_GRADE values.

Table 1. Grading system for coded transcript variables

CRS_GRADE	Corresponding letter grade	CRS_GRADE	Corresponding letter grade
01	A+	11	D
02	A	12	D-
03	A-	13	F
04	B+	14	Pass, satisfactory or credit
05	B	15	Unsatisfactory or no credit
06	B-	16	Withdrew or dropped course
07	C+	17	Incomplete
08	C	18	Non-graded course or audit
09	C-	19	Blank, no grade provided
10	D+	20	Unrecodable grade

In addition to the standardized grade variable, survey staff created a variable for each course called CRS_GRADE_RECODE_STATUS. This variable provides information on how the CRS_GRADE variable was created from the information provided by the school. The values of the recoding status variable are listed in Table 2.

Table 2. Values for CRS_GRADE_RECODE_STATUS

CRS_GRADE_RECODE_STATUS	Recoding Status
0	Directly recoded
1	Recoded using grade specifications of own school
2	Recoded using standard grade specifications
3	Uncodable grade

Each standardized grade was assigned using one of the following four methods:

1. **The transcript reported letter grades using the system in Table 1 above.** All letter grades were directly assigned to the corresponding standardized grade in Table 1. Letters that could not be classified into one of the categories 1-19 were considered to be unrecodable and included in category 20. In the cases where the CRS_GRADE variable was recoded directly from the grade on the transcript, CRS_GRADE_RECODE_STATUS was assigned a value of 0.

2. **The school used numeric grades and provided grading specifications on the one-page Transcript Cover Sheet.** For these respondents, numeric grades were converted to standardized grades using the grading specifications provided by the school. For example, if the numeric grade fell within the range for an 'A' as specified by that particular school, it was assigned to category 02. Fewer than 10% of schools provided multiple grading specifications; in all cases, the primary specifications were used. Due to the possibility of transcription errors, numeric grades below 15 were considered to be unrecodable when the minimum passing grade was higher than 15. For all cases where the CRS_GRADE variable was recoded from the transcript using the school's own grading specifications, CRS_GRADE_RECODE_STATUS was assigned a value of 1.

3. **The school used letter grades of a type different than those shown in Table 1.** Grades of 'G' were classified as 05, 'O' and 'E' as 02, and 'O+' and 'E+' as 01. CRS_GRADE_RECODE_STATUS was assigned a value of 2. If the grades could not be recoded, then CRS_GRADE was assigned a value of 20 and CRS_GRADE_RECODE_STATUS was assigned to 3.

4. **The school used numeric grades and did not provide grading specifications.** The means of the upper and lower limits of the grading systems across all schools were used to construct the standard grading system shown in Table 3. If the school did not specify its grading specifications, numeric grades (and numeric grades with a qualifier attached) were recoded based on this standard system. Once again, to take into account the possibility of transcription errors, numeric grades below 15 were considered to be unrecodable. CRS_GRADE_RECODE_STATUS was assigned a value of 2 when recoding was done using the standard grade specifications. If the grades could not be recoded, then CRS_GRADE was given a value of 20 and CRS_GRADE_RECODE_STATUS was coded as 3.

Table 3. Standard numeric grading system

Lower limit	Upper limit	CRS_GRADE
91	100	02
82	Less than 91	05
73	Less than 82	08
65	Less than 73	11
15	Less than 65	13

NOTES ON TRANSCRIPT SURVEY VARIABLES

Transcript Survey variables can be found on the NLSY97 Event History CD by browsing through the "transcript survey" area of interest or by searching for question names with the prefix TRANS_. This section provides supplemental notes not included in the codebook about individual transcript variables.

Table 4. Additional information about Transcript Survey variables

Question name (all begin with TRANS_) and reference number(s)	Variable title and notes
CRS_CODE.xx R97128.-R97251.	**Course Code xx** Courses are numbered approximately as they appeared on the high school transcript. The course number has no content except to identify the course uniquely. Because terms are numbered approximately chronologically, course numbers and term numbers may not increase in tandem. Information on the Secondary School Taxonomy – Revised (SST-R) is available on the National Center for Education Statistics website, http://nces.ed.gov. **Note that researchers must use the crosswalk provided at the end of this appendix to compare the NLSY97 course codes to the SST-R.**
CRS_CREDIT.xx R97252.-R97375.	**Credits Earned for Course xx** Credits earned are listed in the units provided by the school and are not necessarily comparable across schools. Some schools may award 1 credit for a one-year course, while others award credits according to the number of hours per week that the course meets. Course credits within a term may not sum to the credits earned for the term due to additional credits from non-coursework activities (e.g., community service, yearbook, etc.), or because of credit accumulation rules that involve multiple courses. For example, all religion courses may be worth .5 credits, but a student may face a ceiling of 2 credits earned from religion courses across all high school terms.
CRS_GRADE_RECODE _STATUS.xx R97376.-R97499.	**Recoding Status of Grade for Course xx** See "Course Grades" discussion under Data Entry and Processing above.
CRS_GRADE.xx R97500.-R97623.	**Recoded Quality Grade xx** See "Course Grades" discussion under Data Entry and Processing above.
CRS_TERM_NU.xx R97624.-R97747.	**Course Term Number xx** Because terms are numbered approximately chronologically while courses are numbered approximately as they appeared on the transcript, course numbers and term numbers may not increase in tandem. The term number has no content except to identify the term uniquely.
TERM_CREDIT.xx R97748.-R97765.	**Credits Earned in Term xx** Credits earned are listed in the units provided by the school and are not necessarily comparable across schools. Some schools may award 1 credit for a one-year course, while others award credits according to the number of hours per week that the course meets. Course credits within a term may not sum to the credits earned for the term due to additional credits from non-coursework activities (e.g., community service, yearbook, etc.), or because of credit accumulation rules that involve multiple courses. For example, all religion courses may be worth .5 credits, but a student may face a ceiling of 2 credits earned from religion courses across all high school terms.

Table 4. Additional information about Transcript Survey variables

Question name (all begin with TRANS_) and reference number(s)	Variable title and notes
TERM_START_DATE.xx R97856.-R97873.01	**Month, Year Term Started xx** Calendar month and year listed on transcript for start of term. If only one date was associated with a term, that date appears in TERM_END_DATE.xx. Term dates may overlap if a youth transferred from one school to another mid-session, or in rare cases, with simultaneous enrollment in a vocational school or community college.
TERM_END_DATE.xx R97766.-R97783.01	**Month, Year Term Ended xx** Calendar month and year listed on transcript for end of term. If only one date was associated with a term, that date appears as the end date. Term dates may overlap if a youth transferred from one school to another mid-session, or in rare cases, with simultaneous enrollment in a vocational school or community college.
TERM_GRADE.xx R97784.-R97801.	**Grade Level for Term xx** The grade level (e.g., 10, 11, etc.) in which the youth was enrolled during term xx.
TERM_SCH_NU.xx R97802.-R97819.	**School Number for Term xx** The ID of the school in which the youth was enrolled in term xx. Corresponds only to variables SCH_CAT.xx and not to other school IDs in the NLSY97 youth data. School number 01 indicates the school from which the transcript was received. A school number greater than 01 indicates transferred coursework
TERM_SEASON.xx R97820.-R97837.	**Term Season xx** Calendar season or other term designation of term xx.
TERM_YEAR.xx R97838.-R97855.01	**Term Year xx** Calendar year of term xx designation. May not match TERM_END_DATE.xx or TERM_START_DATE.xx, as in Fall 2000 term ending in January 2001.
SCH_CAT.xx R97874.-R97885.	**Course Catalog Received xx** Marked 'yes' if a course catalog was available from the school during the course coding process. May indicate higher reliability of SST-R code assigned in CRS_CODE.xx. Can be linked to CRS_CODE.xx through term number of course (CRS_TERM_NU.xx) and school number of term (TERM_SCH_NU.xx). School numbers link only to TERM_SCH_NU.xx variable and not to other school IDs in the NLSY97 youth data. School number 01 indicates the school from which the transcript was received, generally the most recent school of enrollment. A school number greater than 01 indicates transferred coursework.
SPECIAL_ED R97886.	**Participated in Special Ed** Marked yes if sampled school indicated on Student Request list that youth was enrolled in special education courses. Pertains to School 01.
BILING_ED R97887.	**Participated in Bilingual Ed** Marked yes if sampled school indicated on Student Request list that youth was enrolled in bilingual education courses. Pertains to School 01.
GIFTED_CRS R97888.	**Participated in Gifted Courses Program** Marked yes if sampled school indicated on Student Request list that youth was enrolled in a gifted courses program. Pertains to School 01.
TERM_TOTAL R97889.	**Total Number of Terms Reported** Total number of terms reported for youth across all schools. Equal to the maximum xx for which TERM_xx variables will have non-missing data.

Table 4. Additional information about Transcript Survey variables

Question name (all begin with TRANS_) and reference number(s)	Variable title and notes
SCH_START_DATE R97890.-R97890.01	**Month, Year Term Started** Calendar month and year in which school shows student as first enrolled. Pertains to School 01.
SCH_END_DATE R97891.-R97891.01	**Month, Year Term Ended** Calendar month and year in which school shows student as last enrolled. Pertains to School 01.
AB_AYxxxx R97892.-R97899.	**Number of Absences in Academic Year xxxx** Total absences in each academic year if youth was enrolled during that school year. For example, variable AB_AY1992 refers to absences in academic year 1992–93. May have been reported annually or summed from term-level data. Pertains to School 01.
AB-MISS R97900.	**Number of Absences if Year Not Assigned** Total absences for youth if absences are not classified by attendance year. Pertains to School 01.
TARDY_AYxxxx R97901.-R97908.	**Number of Tardies in Academic Year xxxx** Total tardies in each academic year if youth was enrolled during that school year. For example, variable TARDY_AY1992 refers to tardies in academic year 1992–93. May have been reported annually or summed from term-level data. Pertains to School 01.
TARDY_MISS R97909.	**Number of Tardies if Year Not Assigned** Total tardies for youth if tardies are not classified by attendance year. Pertains to School 01.
FLAG_MISS_AB_AYxxxx R97910.-R97917.	**Enrolled, Missing Absences in Academic Yr xxxx** Flag indicating that youth was enrolled in an academic year but was missing absence information for that year. For example, FLAG_MISS_AB_AY1992 refers to academic year 1992–93. Pertains to School 01.
FLAG_MISS_TARDY _AYxxxx R97918.-R97925.	**Enrolled, Missing Tardies in Academic Yr xxxx** Flag indicating that youth was enrolled in an academic year but was missing tardy information for that year. For example, FLAG_MISS_TARDY_AY1992 refers to academic year 1992–93. Pertains to School 01.
AT_SCH R97926.	**Has R Left School** School's report of youth's enrollment status in spring 2000. Pertains to School 01.
LEFT_DATE RR97927.-R97927.01	**Month, Year Left School** Calendar month and year in which school assigned non-enrollment status to students who have left school. May differ from SCH_END_DATE because of incomplete requirements that delayed graduation beyond the final term in which student enrolled in courses. May also differ if school has a lag period in which students are not considered to have dropped out, or if a student who transfers out mid-session is recorded as enrolled until the end of that session. Pertains to School 01.
LEFT_REASON R97928.	**Reason Left School** School's report of student's departure status for students who have left school. Pertains to School 01.

Table 4. Additional information about Transcript Survey variables

Question name (all begin with TRANS_) and reference number(s)	Variable title and notes
GPA R97929.	**GPA for Last Year** Grade-point average as calculated by the school in its metric for last year of youth's enrollment. May not match GPA calculated using CRS_GRADE.xx values due to conversion of grades to uniform scale, weighting procedures at school, or other school-specific GPA calculations (e.g., physical education courses do not contribute to academic GPA). Pertains to School 01.
CLASS_RANK R97930.	**Class Rank for Last Year** Youth's rank in class for last year of enrollment. Pertains to School 01.
CLASS_SIZE R97931.	**Class Size Category for Last Year** Categorical variable denoting size of youth's class (grade level) during last year of enrollment. Pertains to School 01.
PSAT_MATH R97932.	**PSAT Math Score** Standardized PSAT math score for youth's last administration of PSAT.
PSAT_VERB R97933.	**PSAT Verbal Score** Standardized PSAT verbal score for youth's last administration of PSAT.
ACT_COMP R97934.	**Composite ACT Score** Standardized ACT composite score for youth's last administration of ACT. Entered directly from transcript, may not correspond to sum of component scores.
ACT_ENG R97935.	**ACT English Score** Standardized ACT English score for youth's last administration of ACT. Entered directly from transcript, may not sum with other components to composite score.
ACT_MATH R97936.	**ACT Math Score** Standardized ACT Math score for youth's last administration of ACT. Entered directly from transcript, may not sum with other components to composite score.
ACT_READ R97937.	**ACT Reading Score** Standardized ACT Reading score for youth's last administration of ACT. Entered directly from transcript, may not sum with other components to composite score.
SAT_VERBAL R97938.	**SAT Verbal Score** Standardized SAT Verbal score for youth's last administration of SAT.
SAT_MATH R97939.	**SAT Math Score** Standardized SAT Math score for youth's last administration of SAT.
SAT_DATE R97940.-R97940.01	**Month, Year SAT Was Taken** Month and year of youth's last administration of SAT.
AP_BIO R97941.	**AP Biology Score** Highest test score for an Advanced Placement biology exam.
AP_CALC R97942.	**AP Calculus Score** Highest test score for an Advanced Placement calculus exam.
AP_CHEM R97943.	**AP Chemistry Score** Highest test score for an Advanced Placement chemistry exam.
AP_ENG R97944.	**AP English Score** Highest test score for an Advanced Placement English exam.

Table 4. Additional information about Transcript Survey variables

Question name (all begin with TRANS_) and reference number(s)	Variable title and notes
AP_HIST_EU R97945.	**AP European History Score** Highest test score for an Advanced Placement European history exam.
AP_GOVT R97946.	**AP Government Score** Highest test score for an Advanced Placement government exam.
AP_PHYS R97947.	**AP Physics Score** Highest test score for an Advanced Placement physics exam.
AP_PSYCH R97948.	**AP Psychology Score** Highest test score for an Advanced Placement psychology exam.
AP_SPAN R97949.	**AP Spanish Score** Highest test score for an Advanced Placement Spanish exam.
AP_HIST_US R97950.	**AP U.S. History Score** Highest test score for an Advanced Placement U.S. history exam.
OTH_AP1 R97959.	**Number Other AP 1** Total number of other Advanced Placement exams on which youth received a score of 1. May include additional exams in subjects listed above.
OTH_AP2 R97960.	**Number Other AP 2** Total number of other Advanced Placement exams on which youth received a score of 2. May include additional exams in subjects listed above.
OTH_AP3 R97961.	**Number Other AP 3** Total number of other Advanced Placement exams on which youth received a score of 3. May include additional exams in subjects listed above.
OTH_AP4 R97962.	**Number Other AP 4** Total number of other Advanced Placement exams on which youth received a score of 4. May include additional exams in subjects listed above.
OTH_AP5 R97963.	**Number Other AP 5** Total number of other Advanced Placement exams on which youth received a score of 5. May include additional exams in subjects listed above.
SATII_BIO R97951.	**SAT II Biology Score** Highest standardized score for an SAT II Biology exam.
SATII_MATH1 R97952.	**SAT II Math I Score** Highest standardized score for an SAT II Math I exam.
SATII_MATH2 R97953.	**SAT II Math II Score** Highest standardized score for an SAT II Math II exam.
SATII_CHEM R97954.	**SAT II Chemistry Score** Highest standardized score for an SAT II Chemistry exam.
SATII_ENG_LIT R97955.	**SAT II English Literature Score** Highest standardized score for an SAT II English Literature exam.
SATII_ENG_WRITE R97956.	**SAT II English Writing Score** Highest standardized score for an SAT II English Writing exam.

Table 4. Additional information about Transcript Survey variables

Question name (all begin with TRANS_) and reference number(s)	Variable title and notes
SATII_HIST_AM R97957.	**SAT II American History and Social Studies Score** Highest standardized score for an SAT II American History and Social Studies exam.
SATII_HIST_WORLD R97958.	**SAT II World History Score** Highest standardized score for an SAT II World History exam.
OTH_SAT1 R97964.	**Number Other SAT 200400** Total number of other SAT II exams on which youth received a score of 200-400. May include additional exams in subjects listed above.
OTH_SAT2 R97965.	**Number Other SAT 401500** Total number of other SAT II exams on which youth received a score of 401-500. May include additional exams in subjects listed above.
OTH_SAT3 R97966.	**Number Other SAT 501600** Total number of other SAT II exams on which youth received a score of 501-600. May include additional exams in subjects listed above.
OTH_SAT4 R97967.	**Number Other SAT 601700** Total number of other SAT II exams on which youth received a score of 601-700. May include additional exams in subjects listed above.
OTH_SAT5 R97968.	**Number Other SAT 701800** Total number of other SAT II exams on which youth received a score of 701-800. May include additional exams in subjects listed above.

CODING INFORMATION FOR COURSE CODE VARIABLES

As part of the NLSY97 Transcript Survey, project staff coded respondents' courses using the revised Secondary School Taxonomy of courses (SST-R). The SST-R codes use a combination of numbers and letters to create a unique identifier for each type and level of high school course content. Because the NLSY97 data base system requires that all data be in numeric form, it was necessary to convert the SST-R codes into numbers for use with the Transcript Survey data. The following table indicates the NLSY97 number assigned for each original SST-R code and course description. These codes were used in variables R97128.–R97251. (TRANS_CRS_CODE.01–.124) in the NLSY97 data set.

Table 5. Crosswalk between NLSY97 codes and SST-R codes

NLSY97 code	SST-R code	Course description
1	1_11A	General Mathematics, ESE/Functional
2	1_11B	General Mathematics, Basic
3	1_11C	General Mathematics, Regular
4	1_11D	General Mathematics, Other
5	1_12A	Consumer Mathematics, ESE/Functional
6	1_12B	Consumer Mathematics, Regular
7	1_13	Pre-Algebra
8	1_14	Algebra 1
9	1_15	Geometry
10	1_16	Algebra 2 through Pre-Calculus TRIG AND ADV MATH
11	1_17A	Advanced Mathematics, Calculus
12	1_17B	AP CALC Advanced Mathematics, AP/IB
13	1_17C	Advanced Mathematics, Other
14	1_18	Unified Mathematics
15	1_19A	Occupationally-Related Mathematics, ESE/Functional
16	1_19B	TECH MATH; Occupationally-Related Mathematics, Regular
17	1_21A	Survey Science, Basic
18	1_21B	Survey Science, Specialized Topics
19	1_21C	Survey Science, Integrated/Unified Topics
20	1_22A	Biological Science, Basic
21	1_22B	Biological Science, Regular
22	1_22C	BIO II; Biological Science, Advanced and Honors
23	1_22D	Biological Science, Specialized Topics
24	1_22E	Biological Science, AP/IB
25	1_23A	Chemistry, Basic
26	1_23B	Chemistry, Regular
27	1_23C	Chemistry, Advanced and Honors
28	1_23D	Chemistry, Specialized Topics
29	1_23E	Chemistry, AP/IB
30	1_24A	Physics, Basic
31	1_24B	Physics, Regular

Table 5. Crosswalk between NLSY97 codes and SST-R codes

NLSY97 code	SST-R code	Course description
32	1_24C	Physics, Advanced and Honors
33	1_24D	Physics, Specialized Topics
34	1_24E	Physics, AP/IB
35	1_25A	Earth Science, Basic
36	1_25B	Earth Science, Regular
37	1_25C	Earth Science, Advanced and Honors
38	1_25D	Earth Science, Specialized Topics
39	1_26A	Physical Science, Basic
40	1_26B	Physical Science, Regular
41	1_26C	Physical Science, Advanced and Honors
42	1_26D	Physical Science, Specialized Topics
43	1_27	Engineering
44	1_31A	English Survey, Language Skills
45	1_31B	English Survey, Grades 7 and 8
46	1_31C1	English Survey, ESE/Functional, Grade 9
47	1_31C2	English Survey, ESE/Functional, Grade 10
48	1_31C3	English Survey, ESE/Functional, Grade 11
49	1_31C4	English Survey, ESE/Functional, Grade 12
50	1_31D1	English Survey, Basic, Grade 9
51	1_31D2	English Survey, Basic, Grade 10
52	1_31D3	English Survey, Basic, Grade 11
53	1_31D4	English Survey, Basic, Grade 12
54	1_31E1	English Survey, Regular, Grade 9
55	1_31E2	English Survey, Regular, Grade 10
56	1_31E3	English Survey, Regular, Grade 11
57	1_31E4	English Survey, Regular, Grade 12
58	1_31F1	English Survey, Advanced and Honors, Grade 9
59	1_31F2	English Survey, Advanced and Honors, Grade 10
60	1_31F3	English Survey, Advanced and Honors, Grade 11
61	1_31F4	English Survey, Advanced and Honors, Grade 12
62	1_31G	AP ENG; English Survey, AP/IB
63	1_32	Literature
64	1_33	Composition and Writing
65	1_34	Speech
66	1_35	English as a Second Language
67	1_41A	American History, Basic
68	1_41B	American History, Regular
69	1_41C	American History, Advanced and Honors
70	1_41D	American History, Specialized Topics
71	1_41E	American History, AP/IB
72	1_42A	World History, Basic
73	1_42B	World History, Regular

Table 5. Crosswalk between NLSY97 codes and SST-R codes

NLSY97 code	SST-R code	Course description
74	1_42C	World History, Advanced and Honors
75	1_42D	World History, Specialized Topics
76	1_42E	World History, AP/IB
77	1_43A	Government & Politics, Basic
78	1_43B	Government & Politics, Regular
79	1_43C	Government & Politics, Advanced and Honors
80	1_43D	Government & Politics, Specialized Topics
81	1_43E	Government & Politics, AP/IB
82	1_44A	Economics, Basic
83	1_44B	Economics, Regular
84	1_44C	Economics, Advanced and Honors
85	1_44D	Economics, Specialized Topics
86	1_44E	Economics, AP/IB
87	1_45A	Behavioral Sciences, Basic
88	1_45B	Behavioral Sciences, Regular
89	1_45C	Behavioral Sciences, Advanced and Honors
90	1_45D	Behavioral Sciences, Specialized Topics
91	1_45E	Behavioral Sciences, AP/IB
92	1_46A	Geography, Basic
93	1_46B	Geography, Regular
94	1_46C	Geography, Advanced and Honors
95	1_46D	Geography, Specialized Topics
96	1_46E	Geography, AP/IB
97	1_47A	Social Science, Humanities, and Other, Basic
98	1_47B	Social Science, Humanities, and Other, Regular
99	1_47C	Social Science, Humanities, and Other, Advanced and Honors
100	1_47D	Social Science, Humanities, and Other, Specialized Topics
101	1_47E	Social Science, Humanities, and Other, AP/IB
102	1_51A	Visual Arts, Basic
103	1_51B	Visual Arts, Regular and Advanced
104	1_51C	Visual Arts, AP/IB
105	1_52A	Music, Basic
106	1_52B	Music, Regular and Advanced
107	1_52C	Music, AP/IB
108	1_53	Dance
109	1_54	Theater Arts
110	1_61A	Spanish, Year 1
111	1_61B	Spanish, Year 2
112	1_61C	Spanish, Year 3
113	1_61D	Spanish, Year 4+
114	1_61E	Spanish, AP/IB
115	1_62A	French, Year 1

Table 5. Crosswalk between NLSY97 codes and SST-R codes

NLSY97 code	SST-R code	Course description
116	1_62B	French, Year 2
117	1_62C	French, Year 3
118	1_62D	French, Year 4+
119	1_62E	French, AP/IB
120	1_63A	German, Year 1
121	1_63B	German, Year 2
122	1_63C	German, Year 3
123	1_63D	German, Year 4+
124	1_63E	German, AP/IB
125	1_64A	Latin, Year 1
126	1_64B	Latin, Year 2
127	1_64C	Latin, Year 3
128	1_64D	Latin, Year 4+
129	1_64E	Latin, AP/IB
130	1_65A	Italian, Year 1
131	1_65B	Italian, Year 2
132	1_65C	Italian, Year 3
133	1_65D	Italian, Year 4+
134	1_65E	Italian, AP/IB
135	1_66A	Other, Year 1
136	1_66B	Other, Year 2
137	1_66C	Other, Year 3
138	1_66D	Other, Year 4+
139	1_66E	Other, AP/IB
140	1_67	Non-English Languages, General/Survey
141	2_AA	Family and Consumer Sciences Education, 1st course
142	2_AB	Family and Consumer Sciences Education, 2nd (or later) courses
143	2_AC	Family and Consumer Sciences Education, Specialty courses
144	2_B1	GLMP, Basic Keyboarding/Typewriting
145	2_B2	GLMP, Industrial Arts
146	2_B3	GLMP, Career Preparation/General Work Experience
147	2_B4	GLMP, Technology Education
148	2_B5	GLMP, Other
149	2_C01A	Agriculture and Renewable Resources, 1st course
150	2_C01B	Agriculture and Renewable Resources, 2nd (or later) courses
151	2_C01C	Agriculture and Renewable Resources, Specialty courses
152	2_C01D	Agriculture and Renewable Resources, Co-op/Work Experience
153	2_C021A	Business Management, 1st course
154	2_C021B	Business Management, 2nd (or later) courses
155	2_C021C	Business Management, Specialty courses
156	2_C021D	Business Management, Co-op/Work Experience
157	2_C022A	Business Services, 1st course

Table 5. Crosswalk between NLSY97 codes and SST-R codes

NLSY97 code	SST-R code	Course description
158	2_C022B	Business Services, 2nd (or later) courses
159	2_C022C	Business Services, Specialty courses
160	2_C022D	Business Services, Co-op/Work Experience
161	2_C03A	Marketing and Distribution, 1st course
162	2_C03B	Marketing and Distribution, 2nd (or later) courses
163	2_C03C	Marketing and Distribution, Specialty courses
164	2_C03D	Marketing and Distribution, Co-op/Work Experience
165	2_C04A	Health Care, 1st course
166	2_C04B	Health Care, 2nd (or later) courses
167	2_C04C	Health Care, Specialty courses
168	2_C04D	Health Care, Co-op/Work Experience
169	2_C05A	Public and Protective Services, 1st course
170	2_C05B	Public and Protective Services, 2nd (or later) courses
171	2_C05C	Public and Protective Services, Specialty courses
172	2_C05D	Public and Protective Services, Co-op/Work Experience
173	2_C061A	T&I, Construction Trades, 1st course
174	2_C061B	T&I, Construction Trades, 2nd (or later) courses
175	2_C061C	T&I, Construction Trades, Specialty courses
176	2_C061D	T&I, Construction Trades, Co-op/Work Experience
177	2_C062A	T&I, Mechanics and Repair, 1st course
178	2_C062B	T&I, Mechanics and Repair, 2nd (or later) courses
179	2_C062C	T&I, Mechanics and Repair, Specialty courses
180	2_C062D	T&I, Mechanics and Repair, Co-op/Work Experience
181	2_C0631A	T&I, Precision Production (Drafting/Graphics/Printing), 1st course
182	2_C0631B	T&I, Precision Production (Drafting/Graphics/Printing), 2nd (or later) courses
183	2_C0631C	T&I, Precision Production (Drafting/Graphics/Printing), Specialty courses
184	2_C0632A	T&I, Precision Production (Metals/Wood/Plastics), 1st course
185	2_C0632B	T&I, Precision Production (Metals/Wood/Plastics), 2nd (or later) courses
186	2_C0632C	T&I, Precision Production (Metals/Wood/Plastics), Specialty courses
187	2_C0633A	T&I, Precision Production (Other), 1st course
188	2_C0633B	T&I, Precision Production (Other), 2nd (or later) courses
189	2_C0633C	T&I, Precision Production (Other), Specialty courses
190	2_C0634	T&I, Precision Production, Co-op/Work Experience
191	2_C064A	T&I, Transportation and Material Moving, 1st course
192	2_C064B	T&I, Transportation and Material Moving, 2nd (or later) courses
193	2_C064C	T&I, Transportation and Material Moving, Specialty courses
194	2_C064D	T&I, Transportation and Material Moving, Co-op/Work Experience
195	2_C071A	Computer Technology, 1st course
196	2_C071BA	Computer Technology, 2nd (or later) courses, non-AP/IB
197	2_C071BB	Computer Technology, 2nd (or later) courses, AP/IB
198	2_C071C	Computer Technology, Specialty courses
199	2_C071D	Computer Technology, Co-op/Work Experience

Table 5. Crosswalk between NLSY97 codes and SST-R codes

NLSY97 code	SST-R code	Course description
200	2_C072A	Communication Technology, 1st course
201	2_C072B	Communication Technology, 2nd (or later) courses
202	2_C072C	Communication Technology, Specialty courses
203	2_C072D	Communication Technology, Co-op/Work Experience
204	2_C073A	Other Technologies, 1st course
205	2_C073B	Other Technologies, 2nd (or later) courses
206	2_C073C	Other Technologies, Specialty courses
207	2_C073D	Other Technologies, Co-op/Work Experience
208	2_C08A	Personal and Other Services, 1st course
209	2_C08B	Personal and Other Services, 2nd (or later) courses
210	2_C08C	Personal and Other Services, Specialty courses
211	2_C08D	Personal and Other Services, Co-op/Work Experience
212	2_C09A	Food Service and Hospitality, 1st course
213	2_C09B	Food Service and Hospitality, 2nd (or later) courses
214	2_C09C	Food Service and Hospitality, Specialty courses
215	2_C09D	Food Service and Hospitality, Co-op/Work Experience
216	2_C10A	Child Care and Education, 1st course
217	2_C10B	Child Care and Education, 2nd (or later) courses
218	2_C10C	Child Care and Education, Specialty courses
219	2_C10D	Child Care and Education, Co-op/Work Experience
220	2_C11	Specific Labor Market Preparation, Unidentified Subject
221	3_1A	Enrichment
222	3_1B	Assistance
223	3_1C	Service
224	3_2	Health, Physical & Recreational Education Credits
225	3_3	Religion and Theology Credits
226	3_4	Military Science Credits
227	4	Special Education Curriculum
228	5_5	Uncodeable