MACRO PL/1
A Programming Language
Developed for Household Survey Analysis

By
Roger L. Wainwright

A paper submitted to the
Computer Science Department in partial fulfillment
of the requirements for the degree
MASTER OF SCIENCE

IOWA STATE UNIVERSITY
AMES, IOWA
1971

The research reported here was supported by funds granted to the Institute
for Research on Poverty at the University of Wisconsin by the Office of
Economic Opportunity pursuant to the provisions of the Economic Opportunity
Act of 1964. The conclusions are the sole responsibility of the author.
MACRO PL/1

I. Rural Family Research Experiment 1
II. Data Management 2
   A. Data Structure 3
   B. Information Retrieval 3
III. Objectives and Desired Capabilities of Macro PL/1 11
IV. Proposed Syntax -- Macro PL/1 13
    A. Preliminary Specifications 13
    B. Macro PL/1 Commands 14
V. Semantics -- Macro PL/1 Commands 17
VI. Sample Macro PL/1 Program 24
I. Rural Family Research Experiment

The Institute for Research on Poverty at the University of Wisconsin, with funding from the Office of Economic Opportunity, is presently conducting a study of the effects of an income maintenance program on the behavior of rural families. Pocahontas and Calhoun counties in Iowa were determined to be quite representative of the rural Midwest, while Duplin county, North Carolina, was determined to be quite representative of the rural South. From these counties approximately 8,344 households were chosen randomly and given a Screening interview. A household is defined as one or more units or families living under one roof. For example, if a husband, wife, and daughter live together along with the wife's grandfather, then two units are living in one household (the grandfather constituting a second unit). Note that unit and family will be used interchangeably throughout this report.

On the basis of the Screening interview about 1,700 of the 8,344 households were selected as potentially eligible to participate in the study. The 1,700 households were given a second interview called Pre-enrollment. On the basis of this interview a subset group was selected to participate in the study. The criteria that were used to select the 1,700 potentially eligible families and to select those families that would finally participate in the study do not concern this report. Presently 825 mainly low and middle income range households are participating in the Rural Family Research project. Households are randomly assigned to either a control group or to one of nine alternative plans corresponding to various minimum guarantees (different percents of the poverty line which are a function of family size only) and tax rates (the amount by which the transfer payment is reduced for each dollar of income earned by the family). The families will receive income supplementation for a period of three years, i. e., twelve quarters. Families participating in this study are interviewed at the end of each quarter on a variety of questions dealing with farming, business, wage jobs, attitudes, health, education, child care costs, etc. Each quarterly interview is called a Quarterly. We are currently in the sixth quarter of the project.

There is a sister experiment to the Rural Family Research project concerning itself with urban families. Cities in New Jersey and Pennsylvania were chosen for this experiment. This paper will not concern itself with the urban project.
The principal purposes of the rural experiment are to measure the effect of alternative tax rates and guaranteed minimums on the work incentive of rural residents and to compare these findings with the work responses of urban families in the New Jersey-Pennsylvania experiment. Other major objectives are to assess the probable impact of a nationwide negative income tax program on rural/urban migration flows and on expenditure patterns. Secondary objectives include measuring the effect of NIT (negative income tax) payments on family health, school performance and social behavior of the children, crime and delinquency rates, family separation, and other attitudinal and behavioral characteristics. The experiment will extend over a fifty-one month period from April 1, 1969, to June 30, 1973.

II. Data Management

Each family is given a unique five digit family number. The first four digits represent the household number, and the least significant digit indicates the unit within the household. The units within a household are numbered with zero as the origin.¹ Each interview, i.e., Screening, Pre-enrollment or Quarterly, is coded for each family into a set of answers. This is because there are many questions in a given interview that have more than one answer associated with them. Thus each interview is considered to yield a set of answers which can be numbered. From the questions in the Screening interview, answers numbered 2 to 208 were obtained for each family. From Pre-enrollment, answers 250 to 1113 were obtained for each family and concatenated to the Screening answers. The reason for the gap in numbering from 208 to 250 will be explained later in the discussion of key variables. With each Quarterly, the number of answers associated with each family will increase. Currently there are about 3,000 answers associated with each family participating in the study. By the time the experiment finishes, 10,000 answers are expected to be accumulated on each family unit.

¹This is a conceptual definition of the family number. In actuality, it is in a slightly different form.
A. Data Structure

For easy reference, a description of each answer has been compiled and organized into a report known as the Instruction Set. Table 1 shows part of the Instruction Set, in particular for answers 199-208, answers 265-267, and answers 347-350. There are essentially two types of answers, fixed and list, denoted as F and L, respectively. A header, denoted by H, may be either a fixed or a list answer. Headers such as answers 206 and 266 contain an integer which says how many lists are contained within them. By convention the first four bytes following a header contain a pointer to the beginning of the first list. The fifth byte is an extent which will be explained later. See Figure 1 for the structure and linking of the data answers described in Table 1. Also by convention, the first four bytes of any list will contain a pointer to the start of the next list associated with the given header. A zero pointer indicates the last list in the chain. The fifth byte from the beginning of any list is the extent.

In Figure 1, 199-208 are fixed answers, 265-267 are list answers, and 347-350 are sublist answers. All fixed answers for any given family are located in the same record called the Fixed record. All list answers, i.e., list and sublist answers, are located in another record called the Variable Length record. Each record is keyed and is located within a Regional (1) dataset, FAMFILE. Note that the words Fixed and Variable Length do not refer to the PL/1 attributes, fixed and variable, associated with records.

The length of each answer, as indicated in Table 1, is the length in bytes. The place in the dataset for fixed answers indicates the byte position of the start of the answer from the beginning of the Fixed record. For lists this indicates the starting byte position relative to the beginning of the list (or sublist).

B. Information Retrieval

ANSDIR (Answer Directory) is the name of a Regional (1) dataset containing the necessary information to retrieve each answer. Each record in ANSDIR is thirty-four bytes long, and the record associated with KEY(n) in ANSDIR contains the information necessary to retrieve answer n. The information depicted for each answer is essentially the same as found in the Instruction Set and is

---


3 Ibid.
<table>
<thead>
<tr>
<th>Answer Number</th>
<th>Type of Answer</th>
<th>Answer Contained Within</th>
<th>Description</th>
<th>Length</th>
<th>Place in the Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>199</td>
<td>F</td>
<td>0</td>
<td>Family number</td>
<td>5</td>
<td>201</td>
</tr>
<tr>
<td>200</td>
<td>F</td>
<td>0</td>
<td>Date of interview--month 8. August 9. September 10. October</td>
<td>2</td>
<td>206</td>
</tr>
<tr>
<td>201</td>
<td>F</td>
<td>0</td>
<td>Date of interview--day</td>
<td>2</td>
<td>208</td>
</tr>
<tr>
<td>202</td>
<td>F</td>
<td>0</td>
<td>Respondent is 1. head of household 2. other filer (head of other family unit within the household) with children 3. other filer (head of other family unit within the household) without children</td>
<td>1</td>
<td>210</td>
</tr>
<tr>
<td>203</td>
<td>F</td>
<td>0</td>
<td>Whether respondent has a spouse 1. yes 2. no</td>
<td>1</td>
<td>211</td>
</tr>
<tr>
<td>204</td>
<td>F</td>
<td>0</td>
<td>Whether or not this interview is a refusal 1. yes 2. no 3. the family moved 4. the respondent died</td>
<td>1</td>
<td>212</td>
</tr>
<tr>
<td>205</td>
<td>F</td>
<td>0</td>
<td>Total number in this family unit 16 and over plus head and spouse</td>
<td>2</td>
<td>213</td>
</tr>
<tr>
<td>206</td>
<td>H</td>
<td>0</td>
<td>Total number in this family unit 16 and over plus head and spouse</td>
<td>2</td>
<td>215</td>
</tr>
<tr>
<td>207</td>
<td>F</td>
<td>0</td>
<td>1969 gross income</td>
<td>6</td>
<td>217</td>
</tr>
<tr>
<td>208</td>
<td>F</td>
<td>0</td>
<td>1969 net business income</td>
<td>6</td>
<td>223</td>
</tr>
<tr>
<td>265</td>
<td>L</td>
<td>206</td>
<td>Family member number</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Answer Number</td>
<td>Type of Answer</td>
<td>Answer Contained Within</td>
<td>Description</td>
<td>Length</td>
<td>Place in the Dataset</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------</td>
<td>--------------------</td>
</tr>
<tr>
<td>266</td>
<td>H</td>
<td>206</td>
<td>Number of jobs respondent has worked at for wages from January 1, 1969, to date of interview</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>267</td>
<td>L</td>
<td>206</td>
<td>Respondent's estimate of gross wages and salary (before deductions) from September 1 through December 31, 1970</td>
<td>4</td>
<td>14</td>
</tr>
<tr>
<td>347</td>
<td>L</td>
<td>266</td>
<td>Kind of work respondent does Six digit code</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>348</td>
<td>L</td>
<td>266</td>
<td>Number of weeks respondent has worked at this job from January 1, 1969, to September 1, 1970</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>349</td>
<td>L</td>
<td>266</td>
<td>Total number of hours per week respondent regularly works at this job</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>350</td>
<td>L</td>
<td>266</td>
<td>Rate of pay per week for this job</td>
<td>3</td>
<td>15</td>
</tr>
</tbody>
</table>
Figure 1
Fixed and Variable Length Records
Showing the Chaining Mechanism in Lists

FIXED Record

Byte positions in this record

201  206  208  210  211  212  213  215  217  221  222  228

01010  9  31  1  1  2  3  2  0136  1  4400  0

199  200  201  202  203  204  205  206  207  208

Answer numbers

(Extent)

Pointer to the beginning of the list in the VARIABLE record

VARIABLE LENGTH Record

Relative byte position within the list

0  4  5  8  13  14

301  1  001  2  400  1500

BYTE 136:  265  266  267

Answer numbers

0  4  5  8  13  14

0  0  002  0  0  100

BYTE 301:  265  266  267

BYTE 400:  347  348  349  350

BYTE 450:  347  348  349  350

BYTE 450:  347  348  349  350

0  4  5  11  13  15

0  0  XXXXX  15  20  60

0  4  5  11  13  15

0  0  XXXXX  15  20  60
illustrated in Figure 2. Currently ANSDIR has about 3,000 records as 3,000 answers have been assembled to date.

Figure 2

Format of the Records in ANSDIR
(Only fields associated with answer retrieval have been shown)

<table>
<thead>
<tr>
<th>L</th>
<th>H</th>
<th>F</th>
<th>Length of answer in bytes</th>
<th>Starting byte position</th>
<th>Answer contained within</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Byte position within the record

In addition to ANSDIR, HMAP (household map) and FMAP (family map) are two more Regional (1) datasets which contain information necessary to retrieve data associated with each family. Figure 3 shows the format of HMAP.

Figure 3

Format of HMAP

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>540</td>
<td>539</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>541</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>102</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>103</td>
<td>750</td>
<td>1011</td>
<td>538</td>
<td>542</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9800</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
HMAP is a 9800 by 5 character array, and each row of HMAP represents one household. The \( n \)th row contains information concerning household \( n \). Fields A through E contain, respectively, row numbers in FMAP (see Figure 4) where information concerning units 0 through 4 can be found. A zero indicates no information. This system restricts the maximum number of units in a household to five. In the majority of the cases, there is only one unit within a household which means field A contains a row number and B, C, D, and E are zero. In Figure 3 household 100 has two units. The necessary information for locating the data, i.e., answers, associated with each unit is in rows 540 and 539 of FMAP, respectively.

FAMFILE is a Regional (1) dataset that contains both Fixed and Variable Length records, as defined previously, for all the families in the experiment. Each record in FAMFILE (Fixed or Variable Length) is 7,000 bytes in length.

Figure 4

Format of FMAP

| Household Number | Unit Number | \( FR_1 \) | \( FR_2 \) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|------------------|-------------|------------|------------|---|---|---|---|---|---|---|---|---|---|---|
| 0                |             |            |            |   |   |   |   |   |   |   |   |   |   |   |
|                  |             |            |            |   |   |   |   |   |   |   |   |   |   |   |
| 538              | 103         | 2          | 467        | 0 | 980| 420| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 539              | 100         | 1          | 950        | 0 | 375| 4911| 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 540              | 100         | 0          | 3190       | 0 | 1956| 691| 2021| 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 541              | 101         | 0          | 1106       | 0 | 97 | 1099| 369 | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 542              | 103         | 3          | 4659       | 0 | 10 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|                  |             |            |            |   |   |   |   |   |   |   |   |   |   |   |
| 1125             |             |            |            |   |   |   |   |   |   |   |   |   |   |   |
FMAP contains a unit identification number, 0 through 4, indicating which unit in the household the information is referring to and the household number which is the link back to HMAP if that should ever be necessary. FR1 contains the key in FAMFILE where the Fixed record for the unit can be found. If 7,000 characters are not enough to hold the fixed answers associated with this unit a second key is specified in FR2, else FR2 is zero. Similarly, for the list answers, up to ten keys can be specified in fields labeled 0 through 9. These fields are called the Extents. A zero in any of the fields, 0 through 9, indicates no information. If you recall, in the discussion of the data structure it was stated that after each header there were five bytes of information. The first four bytes specified the starting byte position in the Variable Length record of the list or sublist, and the fifth byte specified the extent. Actually the fifth byte specified which record the information is in. The fifth byte will contain a value of 0 through 9 indicating which field in FMAP to interrogate to find the key to the record. For example, assume one is retrieving answers associated with family number 5391 (household number 539, unit 1), and a header is interrogated with the five bytes following the header specified as C2501 (position pointer 250, extent 1). From this one should expect to find the beginning of the list starting in byte 250 of the Variable Length record associated with Key(4911) in FAMFILE. Key(4911) is the record specified by extent 1 in FMAP for the above family number (see Figure 4).

ANSDIR, HMAP, and FMAP completely specify the location of any answer for any family in FAMFILE. Due to the size of ANSDIR, HMAP, and FMAP and the frequency of use, they are normally placed in bulk core during any computation.

To conclude this section, a simple example will be given to illustrate the information retrieval system. Suppose it is necessary to determine the estimated amount of gross wages and salary (before deductions) from September 1 through December 31, 1970, for the John L. Smith family. It is first necessary to determine John Smith's family number. This could be done by looking at the interview form he filled out. Assume the family number is 1010. From the Instruction Set it is determined that answer 267 is what we are interested in (see Table 1). The problem can be restated to say that it is necessary to retrieve answer 267 for family 1010. The first step is to retrieve Key(267) of ANSDIR which is shown below.
See Figure 2 for the format of ANSDIR. Since answer 267 is a list answer and is contained within answer 206, it is necessary to retrieve answer 267 by first locating answer 206 for this family. Key(206) of ANSDIR is shown below.

From the above retrieved information, the following quantities can be defined:

Length of 267 = 4, Length of 206 = 2,
Relative Start of 267 = 14 and Start of 206 = 215.

The second step is to determine where in FAMFILE the Fixed and Variable Length records associated with this family are. To do this HMAP is first interrogated. Row 101, field A, will have the desired information as this refers to household 101, unit 0 (family number 1010). Row 101, field A, contains 541 (see Figure 3). This means that family 1010's retrieval information is located in row 541 of FMAP. The FR₁ field of row 541 of FMAP contains 1106 (see Figure 4). This means that the Fixed record associated with family 1010 is associated with Key(1106) of FAMFILE. The next step is to retrieve this Fixed record from FAMFILE. According to the quantities defined above concerning answer 206, answer 206 is located in bytes 215 and 216 of the Fixed record which contains the value of 2 (see Figure 1 which shows the Fixed and Variable Length records associated with family 1010).

The value, 2, indicates two lists which means two answers 267 to retrieve, one in each list. By convention 0136, which is contained in the four bytes
following answer 206, is a pointer to the start of the first list. The extent value of 1 in byte 221 indicates which Variable Length record. From FMAP extent 1 in row 541 indicates that the Variable Length record is associated with Key(1099) in FAMFILE. The next step is to retrieve the Variable Length record. The location of answer 267 in the first list begins in byte 136 + Relative Start of 267 or in byte 150 of the Variable Length record. Answer 267 is a length of four, so bytes 150 through 153 are the location of answer 267 and contain the value 1500. The first four bytes of the list, 136 through 139, point to the start of the second list. The extent is still 1, so the Variable Length record is the same. The location of the second answer 267 begins in byte 301 + Relative Start of 267. The value of the second answer 267 is 100. Byte 301 is zero, so there are no more lists to interrogate. This completes the example, and the result is $1600.

III. Objectives and Desired Capabilities of Macro PL/1

After each interview (Screening, Pre-enrollment or Quarterly) has been given, coded, and placed in FAMFILE, new answers will need to be generated. These new answers are called key answers or key variables and will be generated from the information obtained from the respective interviews. For example one may wish to create a key answer which indicates for each unit how many people earned under $3000 in 1970 and were neither students nor recipients of any government aid. Another example is the problem presented in the last section, where a key answer was generated indicating the sum of answers 267. More realistically, key variables are more complex such as the following: Key Answer = .5(EI - 720 - FT - CC) + UI + 1000(FAP -2) + 300 - BENEFITS. In the above formula EI, FT, CC, UI, FAP, and BENEFITS represent earned income, federal tax (1969), child care cost (annualized), unearned income, number of people in the unit meeting the requirements of the Family Assistance Plan, and the amount of all government aids to the unit, respectively. Each of these quantities, in turn, may require an extensive amount of computation.

In short, from 50 to 200 key answers will be generated for each unit after each interview has been given. Each key answer will be given a unique answer number, and an entry in ANSDIR will be provided to depict its attributes. The reason for the gap in answer numbers, 208 to 250, from Screening to Pre-enrollment can now be explained. It was to allow for key answers on the Screening interview.
For the past nine months, it has been the responsibility of the author to write and maintain the key variable programs associated with the Rural Family Research project. To facilitate ease in writing these programs with the hope of saving money over the next few years, the author has designed a language called Macro PL/1 with the following features and capabilities:

1. The ability to retrieve answers via ANSDIR, HMAP, and FMAP invisible to the user.

2. The ability to compare sets to determine whether one list is a subset or is equal to the other.

3. The ability to perform set operations (as defined in Section IV) on lists.

4. The ability to select from a list. For example, one may wish to select all family members older than sixteen in a particular list.

5. The ability to check arithmetic computation performed by the coders to insure accuracy within a small percentage.

6. The ability to check that similar information requested from Quarterly to Quarterly is accurate within a certain percentage.\footnote{The percentage will vary depending on the magnitude of the answers to be checked.}

7. The ability to check whether the value of a particular answer falls within an accepted range.

8. The ability to perform arithmetic and logical functions.

Macro PL/1 was designed specifically (a) to generate key variables and (b) to check the quality of the data, for the following reasons:

1. The retrieval of answers, in itself, in past programs has constituted nearly one-half of the programming time. This will be greatly reduced in Macro PL/1.

2. Key variables will vary greatly from one interview to another, enough so that new key variable programs will need to be written every quarter. Because of this, routines common to each key variable program need to be generated. These common routines need only be coded once and activated by the Macro PL/1 commands. This is a basic saving in repetition of tasks.

3. In the past, coding errors have occurred in the transferring of information from the interview forms to the coding sheets. In a few cases, errors in the key variable program have caused some new answers to be generated incorrectly. Sometimes, these incorrect answers may be unnoticed for a number of quarters. Also, in a few cases, respondents, i.e., unit heads, do not always respond accurately
to all questions on a given Quarterly due to forgetfulness. The Check and Check Value commands were designed specifically to catch these errors by providing an ability for checking relationships between the data over Quarters.

The types of checks that are made fall into three classes:

(a) Total versus parts. For example, the coders added answers 66 and 67 by hand and created answer 68 as the result. Their work can be checked and it should be exact.

(b) Close quantities. For example, income tax is figured using two different but similar procedures on the income of a given unit. The results should be close, that is, within 10 percent of each other.

(c) Loose quantities. For example, wages from one quarter to the next for a given individual should not change drastically, that is, within 50 percent of each other.

IV. Proposed Syntax--Macro PL/1 in Backus Normal Form.

A. Preliminary Specifications

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<letter> ::= A | B | C | ... X | Y | Z

<letter - AR> ::= B | C | D | ... P | Q | S | T | ... X | Y | Z

<zero or one> ::= 0 | 1

<natural number> ::= <digit> | <natural number> <digit>

<integer> ::= <natural number> | + <natural number> | - <natural number>

<key answer> ::= AN <natural number>

<answer> ::= AN ((<natural number>))

<variable> ::= <letter - AR> | <variable> <letter> | <variable> <digit>

(Note that this mechanism does not allow variables to begin with A or R. Variables beginning with these letters are reserved for the translator.)

<variable or integer> ::= <variable> | <integer>

<factor> ::= <variable> | <answer> | <integer> | (<A.E.>)

<term> ::= <factor> | <term>*<factor> | <term>/<factor>
\[ \langle A.E. \rangle ::= \langle \text{term} \rangle \mid \langle A.E. \rangle + \langle \text{term} \rangle \mid \langle A.E. \rangle - \langle \text{term} \rangle \]
\[ \langle \text{relation} \rangle ::= = \mid \langle \langle \rangle \rangle = \mid \langle \langle \langle \rangle \rangle \rangle = \langle \langle \langle \rangle \rangle \rangle = \langle \langle \langle \rangle \rangle \rangle \]
\[ \langle \text{inclusive A.E.} \rangle ::= \langle A.E. \rangle \mid \langle A.E. \rangle \text{ TO } \langle A.E. \rangle \]
\[ \langle \text{statements} \rangle ::= \text{any Macro PL/1 command (s) and/or PL/1 statement (s)} \]
\[ \langle \text{set operation} \rangle ::= \text{INTERSECT} \mid \text{UNION} \mid \text{SUBTRACT} \]
\[ \langle \text{set A.E.} \rangle ::= \langle \text{key answer} \rangle \mid \langle \text{set A.E.} \rangle \langle \text{set operation} \rangle \langle \text{set A.E.} \rangle \]
\[ \langle \text{range 1} \rangle ::= \langle \text{key answer} \rangle \text{ TO } \langle \text{key answer} \rangle \]
\[ \langle \text{while condition} \rangle ::= \text{WHILE} \langle \text{key answer} \rangle \langle \text{relation} \rangle \langle \text{inclusive integer} \rangle, \ldots, \langle \text{inclusive integer} \rangle; \]
\[ \text{(From one to twenty \langle \text{inclusive integer} \rangle 's are allowed.)} \]

B. Macro PL/1 Commands

1. \[ \langle \text{Range command} \rangle ::= \text{RANGE } (\langle \langle \text{range 1} \rangle \rangle) \mid \text{RANGE } (\langle \langle \text{range 1} \rangle, \langle \text{range 1} \rangle \rangle) \]

2. \[ \langle \text{Do command} \rangle ::= \text{DO\_LIST } (\langle \langle \text{key answer} \rangle \rangle) \mid \text{DO\_LIST } (\langle \langle \text{variable}=\langle \text{variable or integer} \rangle \text{ TO } \langle \text{key answer} \rangle \rangle) \]

3. \[ \langle \text{End list command} \rangle ::= \text{END\_LIST} \]

4. \[ \langle \text{Search command} \rangle ::= \text{SEARCH } (\langle \langle \text{key answer} \rangle \text{ SUBSET } \langle \text{key answer} \rangle \rangle) \mid \text{SEARCH } (\langle \langle \text{key answer} \rangle = \langle \text{key answer} \rangle \rangle) \]

5. \[ \langle \text{Check command} \rangle ::= \text{CHECK } (\langle \langle \text{INTEGER}, \langle \text{A.E.} \rangle = \langle \text{A.E.} \rangle \rangle \text{ WITHIN} \langle \langle \text{variable or integer} \rangle \rangle \langle \text{PERCENT} \rangle) \]

6. \[ \langle \text{Check value command} \rangle ::= \text{CHECK VALUE } (\langle \langle \text{INTEGER}, \langle \text{A.E.} \rangle = \langle \text{inclusive A.E.} \rangle \rangle; \langle \langle \text{inclusive A.E.} \rangle \rangle; \ldots \langle \langle \text{inclusive A.E.} \rangle \rangle) \]
\[ \text{(up to 20 \langle \text{A.E.} \rangle and up to 10 \langle \text{A.E.} \rangle \text{ TO } \langle \text{A.E.} \rangle \text{ are allowed})} \]
7. \(<\text{When command}\> ::= \text{WHEN} ((<\text{A.E.}> <\text{relation}> <\text{inclusive A.E.}>; <\text{inclusive A.E.}>; ...<\text{inclusive A.E.}> \text{ THEN} <\text{statements}>))
\quad \text{(up to 20}<\text{A.E.}>\text{ and up to 10}<\text{A.E.}>\text{ TO}<\text{A.E.}>\text{ are allowed)}

8. \(<\text{Breaks subcommand}\> ::= \text{BREAKS} (<\text{A.E.}>, <\text{integer}>, <\text{integer}>
\quad <\text{natural number}>, <\text{zero or one}>)

9. \(<\text{Assignment command}\> ::= \text{ASSIGN} (((<\text{key answer}> = <\text{A.E.}> ;))

10. \(<\text{Set check command}\> ::= \text{SET\_CHECK} (((<\text{set A.E.}> <\text{subset or} = > <\text{set A.E.}> ;)) |
\quad \text{SET\_CHECK} ((<\text{set A.E.}> = \text{NULL})) |
\quad \text{SET\_CHECK} ((<\text{set A.E.}> = \text{NULL}, <\text{while condition}>)) |
\quad \text{SET\_CHECK} (((<\text{set A.E.}> <\text{subset or} = > <\text{set A.E.}> <\text{while condition}>))

Before the semantics of each command are discussed, it may be wise to digress for a moment to explain how Macro PL/1 will fit into the system. It is intended that future key variable programs be written in PL/1 with these special Macro PL/1 commands used where appropriate. It is obvious that Macro PL/1 commands will not do everything that is needed in a key variable program. This is why PL/1 is used as a basis. In the future PL/1 with Macro PL/1 commands inserted freely will be referred to as the Macro PL/1 Language.

The translator to interpret Macro PL/1 language will in concept scan the input looking for only Macro PL/1 commands and key answers, ignoring everything else. When a Macro command is found, the translator will replace the command with PL/1 statements designed to do what the Macro command specifies. The same is true when the translator scans an<answer>, which may appear in a Macro command or within a PL/1 statement. The semantic interpretation of each Macro command is discussed in Part V. Figure 5 illustrates a flow diagram for the execution of any Macro PL/1 program.
In practice, the function of the translator will be performed by the PL/1 Preprocessor\(^5\) at compile time. There is a preprocessor procedure for each of the ten commands except for the Breaks command. There is also a preprocessor procedure for the interpretation of an `<answer>`. At compile, the preprocessor procedures will replace their respective commands with a procedure call to a run time procedure to perform the function of the Macro command with PL/1 code designed to perform the Macro command. See Section

VI for a sample Macro PL/1 program along with all of the compile and run-time procedures.

V. Semantics--Macro Commands

1. Range Command

Example 1: RANGE ((AN619 TO AN705))

Example 2: RANGE ((AN204 TO AN309, AN512 TO AN917))

Restrictions: Within <range 1> the key answer preceding TO must be less than or equal to the key answer following TO. In addition, blanks must surround TO. In the above examples AN619 <= AN705 and AN204 <= AN309, etc. This command is mandatory and must appear before any other Macro commands in the program if one wishes to write on the dataset. Other Range commands may follow throughout the program with the condition that the new Range command will cancel the effect of the old one. If no Range command appears in the program, the default is RANGE ((ANO TO ANO)).

Semantics: This command generates no PL/1 code. It is used to signal the Macro translator which key variables are going to be generated by the program. All key variables that appear on the left side of the equal sign in an Assignment command are checked to be within the range specified by the Range command. If it is not within range, then an error message is generated and the Assignment command is not performed. For example, if the range is specified as in Example 1 above, and the programmer specified AN269 by accident instead of AN629 in an Assignment command; then answer 269 would be destroyed and answer 629 will reside in the wrong place in the dataset. The purpose of this command is to prevent such a holocaust. In other words, this command assures the user that only those answers specified in the Range command are eligible for updating.

2. Do Command

Example 1: DO_LIST ((AN571))

Example 2: DO_LIST ((I = 1 TO AN571))

Example 3: DO_LIST ((I = J TO AN 571))

Restrictions: The key answers must be headers. If they are not, an error message is generated. In addition, blank(s) must surround TO.

Semantics: PL/1 code will be generated to perform a DO loop. This involves the interrogation of HMAP, FMAP, and ANSDIR to obtain the contents of the specified <key answer>, the record key, and the starting position of the list or sublist. The DO loop is used to execute statements associated with the key
answer and delimited by the DO LIST and END LIST commands repetitively. In Example 1, the range of the DO loop defaults to 1 as the initial value and the value of AN571 as the terminating value. Example 2 has the same range. The difference lies in the fact that the user now has access to the value of the loop variable. In Example 3, the user has further control because the initial value of the loop may be specified.

3. End List Command
   Example: END_LIST
   Restrictions: For every Do command there must be an associated End List command.
   Semantics: The code generated by this command will essentially close the Do loop. The pointer to the beginning of the next list is calculated prior to the ending of the loop.

4. Search Command
   Example 1: SEARCH ((AN756 = AN940))
   Example 2: SEARCH ((AN756 SUBSET AN940))
   Restrictions: The key answers must be list or sublist answers. Blank (s) must surround the equal sign or SUBSET and this command may not appear within a DO_LIST -- END_LIST block.
   Semantics: Each key answer by virtue of being a list or sublist answer determines a set (in the mathematical sense) of answers. In Example 1, the set of AN756 answers is searched against the set of AN940 answers to see if they are equal. If they are not, then an error message is generated indicating which elements caused the set to be unequal. In Example 2, the set of AN756 answers is searched against the set of AN940 answers to see if set AN756 is a subset of set AN940. If it is not, then an error message is generated indicating which elements caused the subset relationship not to hold.

5. Check Command
   Example 1: CHECK ((4, AN((560)) + AN((600)) = AN620 WITHIN 2 PERCENT))
   Example 2: CHECK ((1, AN((560)) * 12 - AN((571)) = AN((620)) - AN((571)) * I WITHIN X PERCENT))
   Restrictions: None
   Semantics: The purpose of this command is to check whether two numerical quantities are equal within a specified error tolerance. The tolerance is based on the given percentage with respect to the expression on the left side of the equal sign. If the quantities are not equal within a specified range, then an error message is generated. The <integer>(first parameter) of the command is printed in the error message to identify which Check command is issuing the message. One application
of this command is cited as follows. AN620 was generated by hand addition by the coders at the time each interview was coded. This command has the capability of checking this prior calculation for accuracy.

6. Check Value Command

Example 1: CKVALUE ((10, AN((106)) = 5; 10; 17; 94.))

Example 2: CKVALUE ((6, AN((106)) - AN((107)) * 12 = 5; 10 TO 17;
AN((108)); AN((110)) TO 1000; 56 * AN((406)) -
(AN((407)) + 10).))

Restrictions: Up to twenty <A,E>'s and up to ten (<A,E> TO <A,E>)'s are allowed. Blank (s) must surround TO. A ',' must terminate the list of range and values that are to be checked, and no other '.' may appear within the command. A ';' must separate the list of ranges and/or values.

Semantics: The code generated for this command will check to see if the <A,E> on the left side of the equal sign is equal to any of the <A,E>'s on the right side of the equal sign or if it is within range of any <inclusive A,E>'s (including the end points) on the right side of the equal sign. An error message is generated if this check fails. The <integer> (first parameter) in this command is used in the same way as the Check command. The integer simply identifies which Check Value command is issuing the error message. The purpose of this command is to have a check on the key variables that are generated as well as the answers that were generated from each interview. For example, if AN106 is a key variable representing the average age of the members of the unit, one can check this answer for serious errors before it is placed onto the dataset by executing: CKVALUE ((AN((106)) = 15 TO 90.))

7. When Command

Example 1: WHEN (( AN((106)) = 5; 10; 17; 94 THEN GO TO LABEL.))

Example 2: WHEN (( AN((106)) - 146 * AN((151)) = 5; 10 TO 17; AN((108))
- 56 TO 562 - AN((110)) * 12 THEN I=G; DO LIST
(AN12) I = I + AN((149)) ; END_LIST . ))

Restrictions: Up to twenty <A,E>'s and up to ten (<A,E> TO <A,E>)'s are allowed. Blank (s) must surround TO and the <relation>. A ';' separates the items in the list but no ';' follows the last <inclusive A,E> before the THEN. A '.' must terminate the <statements> and no other '.' may appear within the command.

Semantics: This command is much like an if statement in PL/I. If the <A,E> on the left side of the relation satisfies the relation with any one of the inclusive<A,E>'s on the right side of the <relation>, then and only then are the <statements> executed.

Example 1: \( I = \text{BREAKS ( AN((105)), -600, 600, 200, 1); } \)

Example 2: \( I = \text{BREAKS ( AN((114)) * 6 -AN((119)) + W, 0, 100, 10, 1); } \)

Restrictions: The left integer must be less than the right integer and the natural number must be less than the sum of the absolute values of the integers. (Note that this command uses a single set of parenthesis.)

Semantics: The three numbers in the command from left to right specify, respectively, the lower bound, the upper bound, and the increment of the desired breaks. For example, the breaks defined by Example 1 are as follows:

<table>
<thead>
<tr>
<th>Breaks</th>
<th>Break Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-600 or less</td>
<td>0</td>
</tr>
<tr>
<td>-599 to -400</td>
<td>1</td>
</tr>
<tr>
<td>-399 to -200</td>
<td>2</td>
</tr>
<tr>
<td>-199 to 0</td>
<td>3</td>
</tr>
<tr>
<td>1 to 200</td>
<td>4</td>
</tr>
<tr>
<td>201 to 400</td>
<td>5</td>
</tr>
<tr>
<td>401 to 600</td>
<td>6</td>
</tr>
<tr>
<td>over 600</td>
<td>7</td>
</tr>
</tbody>
</table>

A zero or one following the first three numbers in the command specifies the type of break associated with zero. A one indicates that zero is not treated as a separate break or category in itself, as shown above. However, if Example 1 is changed to \( I = \text{BREAKS ( AN((105)), -600, 600, 200, 0); } \), then zero is treated as an isolated break in itself, as shown below.

<table>
<thead>
<tr>
<th>Breaks</th>
<th>Break Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-600 or less</td>
<td>0</td>
</tr>
<tr>
<td>-599 to -400</td>
<td>1</td>
</tr>
<tr>
<td>-399 to -200</td>
<td>2</td>
</tr>
<tr>
<td>-199 to -1</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1 to 200</td>
<td>5</td>
</tr>
<tr>
<td>201 to 400</td>
<td>6</td>
</tr>
<tr>
<td>401 to 600</td>
<td>7</td>
</tr>
<tr>
<td>over 600</td>
<td>8</td>
</tr>
</tbody>
</table>

This command will cause the A.E. to be evaluated; and a break value will be determined depending on the break schedule requested, as explained above. The variable on the left of the equal sign will be assigned the break value. For example, if AN105 has the value -370 in Example 1, then I will be assigned the value 2.
9. Assignment Command

Example 1: `ASSIGN (( AN510 = I + J;))`

Example 2: `ASSIGN (( AN510 = AN((450)) + AN((940)));)`

Example 3: `ASSIGN (( AN510 = BREAKS ( AN((500)), 0, 200, 20, 0;));)`

Restrictions: The `<key answer>` on the left side of the equal sign must be in the range specified by the Range command, otherwise, the assignment is not performed.

Semantics: The `<A.E.>` is evaluated, and the result is assigned to the `<key answer>`.

10. Set Check Command

Example 1: `SET_CHECK (( AN105 UNION AN106 = AN150;))`

Example 2: `SET_CHECK (( AN105 SUBSET (AN150 SUBTRACT AN151) UNION AN152;))`

Example 3: `SET_CHECK ((AN105 INTERSECT (AN106 SUBTRACT AN190) = NULL;))`

Example 4: `SET_CHECK ((AN400 = AN401 UNION AN402 WHILE AN403 = 25;))`

Restrictions: Blanks must surround the `<set operation>`'s and the equal sign.

Semantics: The purpose of this command is to have the ability to do set comparisons of a more sophisticated nature than what is allowed in the Search Command. Each `<key answer>` is considered to be a set of values. If the key answer is a fixed answer, then this represents a singleton set, namely, the value of the answer. However, for list and sublist answers, there may be as many elements in the set as there are values for the answer. In Example 1, a check is made to see if the sets of answers AN105 UNION AN106 is equal to the set of answers of AN150. In Example 2, a check is made to see if set AN105 is a subset of the union of sets AN152 and the set resulting from set AN150 subtracting set AN151. In Example 3, a check is made to see if set AN105 intersected with the set resulting from set AN106 subtract AN190 is NULL or the empty set. Example 4 is a conditional check. The elements of the union of sets AN401 and AN402 are only those elements for which the associated AN403 <= 25. This is illustrated below. Assume AN400 through AN403 are list answers of the header and there are five lists.
AN400  AN401  AN402  AN403
LIST 1  John  Mary  Joyce  5
LIST 2  Mary  Joyce  Joe  15
LIST 3  Joe  John  Sam  19
LIST 4  Sam  George  Sandy  45
LIST 5  Joyce  Joe  Mary  25

Set AN400 = {John, Mary, Joe, Sam, Joyce}.
Set AN401 UNION AN402 = {John, Mary, Joe, Sam, Joyce, Sandy, George}.
Set AN401 UNION AN402, while AN403 <= 25 = {John, Mary, Joe, Sam, Joyce}.

If the check is not satisfied in a Set Check command, then an error message is generated describing the error.

The interpretation of an <answer> in Macro PL/1 is implied by context. For a fixed answer, the <answer> simply refers to the one answer in all cases. If an <answer> is a list or sublist answer and does not appear within a DO_LIST - END_LIST block, then the sum of the list or sublist answers is implied. If a list answer appears within a DO_LIST - END_LIST block, then this simply refers to the answer within the current list, while a sublist answer again implies the sum. As for a list answer in a single DO_LIST - END_LIST block, a sublist answer appearing within a double nested DO_LIST - END_LIST block simply refers to the answer within the current sublist. Recall that a simple example was given to determine the sum of answer 267 for particular family at the end of Section II. This can be accomplished by the following statements:

\[
J = 0;
DO_LIST ((AN206))
J = J + AN((267))
END_LIST;
\]

J will contain 1600 (see Figure 1).

The same result can be obtained in one command: \( J = AN((267)); \). Since AN267 is a list answer, the sum is implied. The sublist answer, AN350, can be summed by the single command: \( J = AN((350)); \) or by the following set of commands:
23

\begin{verbatim}
J = 0;
DO_LIST ((AN206)).
J = J + AN(350))
END_LIST

Or by:

J = 0;
DO_LIST ((AN206))
DO_LIST ((AN266))
J = J + AN(350));
END_LIST
END_LIST
\end{verbatim}

In the author's opinion, the development of a special purpose language such as Macro FL/1 to aid in the task of repeatedly generating and manipulating large volumes of data will represent a substantial savings to the project.
COMPILE-TIME MACRO PROCESSOR
MACRO SOURCE: LISTING

1  (SUBRG, STRG):
2  /* IOWAKEY - ROGER L. WAINWRIGHT : PROGRAM TO GENERATE IOWA
3    KEY VARIABLES USING PL/1 AND MACRO PL/1 */
4  IOWAKEY: PROCEDURE OPTIONS( MAIN );
5  % DECLARE CHECK ENTRY (CHARACTER) RETURNS (CHARACTER);
6  % DECLARE AN ENTRY (CHARACTER) RETURNS (CHARACTER);
7  % DECLARE DO_LIST ENTRY (CHARACTER) RETURNS (CHARACTER);
8  % DECLARE SEARCH ENTRY (CHARACTER) RETURNS (CHARACTER);
9  % DECLARE CKVALUE ENTRY (CHARACTER) RETURNS (CHARACTER);
10  % DECLARE PREADN CHARACTER;
11  % DECLARE REED ENTRY (CHARACTER) RETURNS (CHARACTER);
12  % DECLARE ASSIGN ENTRY (CHARACTER) RETURNS (CHARACTER);
13  % DECLARE WHEN ENTRY (CHARACTER) RETURNS (CHARACTER);
14  % DECLARE RANGE ENTRY (CHARACTER) RETURNS (CHARACTER);
15  % DECLARE END_LIST CHARACTER;
16  % DECLARE (RFAMFILE, RDIRCTRY, RHOUSEMP) CHARACTER;
17  % DCL RDINFO CHARACTER;
18  % RDINFO = 'RDINFO';
19  % RFAMFILE = 'RFAMFILE';
20  % RDIRCTRY = 'RDIRCTRY';
21  % RHOUSEMP = 'RHHOUSEMP';
22  /* ******************************************* PREPROCESSOR END_LIST ******************** */
23  /* FORMAT: END_LIST */
24  */
25  % END_LIST = 'IF RNEST(1) = RANSWER & RNEST(2) = RANSWER THEN DO;
26  PUT EDIT ( 'END_LIST ERROR; RANSWER = ** RANSWER ( 'SKIP A , F(6) );
27  GO TO
28  RREAD: END; RSTLIST ( RANSWER ) = SUBSTR( RFVFILE ( RETX( RANSWER ) ) ) , RSTLIST
29  ( RANSWER ) +4,1); END; IF RNEST(1) = RANSWER THEN RNEST(1) = 0; IF RNEST
30  (2) = RANSWER THEN DO; RNEST(2) = 0; RANSWER = RNEST(1); END; *;
31  */
32  /* ******************************************* PREPROCESSOR DO_LIST ******************** */
33  /* FORMAT: DO_LIST ( (AN547)) OR
34  DO_LIST ( ( <VARIABLE> = <VARIABLE OR INTEGER> TO AN547 ) )
35  RESTRCTIONS: ZERO OR MORE BLANKS MAY PRECEDE OR FOLLOW '* ', ' OR ',
36  OR <VARIABLE> OR ' =' OR <VARIABLE OR INTEGER> ';
37  BLANK(S) MUST SURROUND 'TO' AND NO BLANKS MAY APPEAR
38  WITHIN THE KEY ANSWERS. */
39  */
40  $DO_LIST: PROCEDURE ( TEXT ) RETURNS ( CHARACTER );
41  /* IN EFFECT THIS PROCEDURE SETS UP THREE PARAMETERS
42  RSTLIST, REXT, RHEAD FOR THE LOOP */
43  DECLARE ( TEXT, DLOOP, A1, X0, X1, X2, X3, X4, FINAL ) CHARACTER;
44  DECLARE ( R1, R2, SWITCH ) FIXED;
45  SWITCH = 0; X0 = ' '; FINAL = ' *';
46  DO R2 = 1 TO 29; IF SUBSTR ( TEXT, R2, 1 ) = '*' THEN GO TO LO; END;
47  LO: DO R1 = 1 TO 99; IF SUBSTR ( TEXT, R1, 2 ) = 'AN' THEN GO TO LI;
48  IF SUBSTR ( TEXT, R1, 1 ) = '*' THEN SWITCH = 1; END;
MACRO SOURCE LISTING

50 L1: R1 = R1 + 1;
51 /* R1 POINTS TO THE 'N' OF 'AN' */
52 IF SWITCH=1 THEN DLOOP= SUBSTR(TEXT,R2+1,R1-R2-2);
53 DO R2=R1+1 TO 99;
54 IF SUBSTR(TEXT,R2,1)="ilmiş"
55 STRUBR(TEXT,R2,1)=1 THEN GO TO L2;
56 END;
57 L2: A1= SUBSTR(TEXT,R1+1,R2-R1-1);
58 IF SWITCH=0 THEN DO:
59 X4= DO R1=1 TO A1 IF RHEAD(R1); IF A1(A1 R1)=" ";
60 ELSE X4= DO R1=1 TO OLOOP RHEAD(R1=1);
61 X1" " RANSWER = " " A1" "
62 X2=" CALL RDO(RANSWER, 0,NEST,SRSTLIST,REXT,RHEAD,RFAMREC,
63 RFIXLIMIT,RDIRCTRY,REDDUCE);
64 IF REDDUCE=1 THEN GO TO RREAD;
65 X3=" IF RERCODE=1 THEN GO TO RREAD;
66 FINAL= FINAL || X0 || X1 || X2 || X3 || X4;
67 RETURN(FINAL);
68 %END D0_LIST;
69
70 /* ******************* PREPROCESSOR AN ******************* */
71 /* FORMAT: AN((647)) OR AN((647C)) */
72 RESTRICTIONS: (1) C INDICATES THE ANSWER WILL BE RETURNED IN
73 CHARACTER FORM ELSE FIXED BIN(31,0) IS ASSUMED.
74 (2) ANY NUMBER OF BLANKS (ZERO OR MORE) MAY PRECEDE
75 OR FOLLOW 'AN' OR IF 'I' OR 'T'
76 (3) THE KEY ANSWER MAY NOT CONTAIN ANY BLANKS.
77 %AN: PROCEDURE (TEXT) RETURNS(CHARACTER);
78 /* THIS PREPROCESSOR PROCEDURE WILL CREATE ONE OF TWO
79 FUNCTION PROCEDURE CALLS, ANSW OR ANSWC. */
80 DECLARE(FRONT,FRONTC,ANS,BACK,FINAL,TEXT,ITEM, ICHARACTER);
81 DECLARE(I,J,FIXED);
82 BACK='RDIRCTRY,' NEST,SRSTLIST,REXT,RFAMREC,RFIXLIMIT)" :
83 DO I=1 TO 29; IF SUBSTR(TEXT,I)=" " THEN GO TO L0; END;
84 L0: /* I POINTS TO " " */
85 DO J=1 TO 99; /* SKIP OVER THE BLANKS BETWEEN '* ' AND THE INTEGER*/
86 IF SUBSTR(TEXT,J)=" " THEN GO TO L1; END;
87 L1: I=J; /* I POINTS TO THE FIRST INTEGER */
88 DO J=I TO 99; IF ITEM="C" THEN GO TO FOUND;
89 IF ITEM="C" THEN GO TO NOC; END;
90 NOC: ANSW= SUBSTR(TEXT,J-1); END;
91 FINAL = FRONT || ANS || BACK;
92 GO TO RETFIN;
93 FOUND: ANSW= SUBSTR(TEXT,J-1);
94 FINAL = FRONTC || ANS || BACK;
95 RETFIN: RETURN(FINAL);
96 % END AN;
97 /* ******************* PREPROCESSOR REED ******************* */
98
MACRO SOURCE2 LISTING

99  PROCEDURE(TEXT) RETURNS(CHARACTER):
100  /* THIS ROUTINE WILL READ FROM FILE(RODIRCTRY) THE ANSWER
101  SPECIFIED BY THE PARAMETER INPUT IF IT HAS NOT BEEN DONE BEFORE.
102  IF THIS HAS BEEN DONE ALREADY THEN RDlixirFLO IS ACCESSED TO FIND THE
103  INFORMATION
104  */
105  DCL {TEXT,FINAL } CHARACTER;
106  FINAL = 'REEDX = {"{TEXT}" }'' ;
107  IF RDlixirFLO( REEDX ) = 0 THEN DO; READ FILE(RODIRCTRY)
108  INTO(RANSIDIR KEY( REEX ) ; RDIRSPAC = RDIRSPAC+1;
109  RDIRINFO(RDIRSPAC)= RANSIDIR; RDIRMAP( REEDX ) = RDIRSPAC;
110  END;
111  ELSE RANSIDIR = RDIRINFO(RDIRMAP( REEDX ));
112  RETURN(FINAL);
113  END REED;
114  /* ************** PREPROCESSOR RANGE **********************/
115  /* FORMAT: RANGE ((AN640 TO AN720) OR
116  RANGE ((AN640 TO AN720; AN94 TO AN102))
117  RESTRICTIONS: (1) BLANKS MUST SURROUND 'TO'.
118   (2) ANY NUMBER OF BLANKS (ZERO OR MORE) MAY FOLLOW
119   'RANGE' OR PRECEDE OF FOLLOW EACH '{' OR '}'.'
120   (3) THERE CAN BE NO BLANKS WITHIN THE KEY ANSWERS */
121  RANGE: PROCEDURE(TEXT) RETURNS(CHARACTER):
122  /* THE PREPROCESSOR FUNCTION RANGE WILL SET THE RANGE OF THE
123  ELIGIBLE KEY ANSWERS TO BE UPDATED */
124  DECLARE(ASSIGNMENTS,ITEM,A1,A2,X1,X2,TEXT ) CHARACTER;
125  DECLARE(R1,R2,TL,T2 ) FIXED;
126  X1=' RAN(1) = '; X2=' RAN(2) = '; ASSIGNMENTS='';
127  DO R2=1 TO 29; IF SUBSTR(TEXT,R2,1)="{ " THEN GO TO L0; END;
128  L0: /* R2 POINTS TO ' ' */
129  DO R1= R2 +1 TO 99; IF SUBSTR(TEXT,R1,2)="AN" THEN GO TO L1; END;
130  L1: /* R1 = R1 +1;
131  /* R1 POINTS TO N OF THE FIRST 'AN' */ DO R2= R1+1 TO 99;
132  IF SUBSTR(TEXT,R2,1)="{ " THEN GO TO L2; END;
133  L2: A1= SUBSTR(TEXT,R1+1,R2-R1);
134  DO R1= R2 TO 99; IF SUBSTR(TEXT,R1,2)="AN" THEN GO TO L3; END;
135  L3: R1= R1 +1;
136  /* R1 POINTS TO N OF THE SECOND 'AN' */ DO R2= R1+1 TO 99;
137  IF ITEM= " " | ITEM= " " | ITEM= " " | ITEM= " " THEN GO TO L4;
138  THEN GO TO L4;
139  L4: A2= SUBSTR(TEXT,R1+1,R2-R1-1);
140  T1=A1; T2=A2; IF Y1>T2 THEN DO R1= R2 TO 99; IF ITEM= " " THEN GO TO L6;
141  ASSIGNMENTS= ASSIGNMENTS || X1 || A1 || " " || X2 || A2 || " " ||
142  /* CHECK IF THE THE RANGE COMMAND HAS ONE OR TWO RANGES */
143  THEN GO TO L5; IF ITEM= " " THEN GO TO L6; END;
144  GO TO ERR;
(SUBRG, STRG):

MACRO SOURCE2 LISTING

L5: RETURN(ASSIGNMENTS);
L6: X1 = 'RAN(3) = ' ; X2 = ' RAN(4) = ' ; R2 = R1; GO TO L0;
ERR: ASSIGNMENTS = ****** RANGE COMMAND ERROR ******* ;
GO TO L5;
% END RANGE;

/*  ****************************************************** PREPROCESSOR PREASN ******************** */

% PREASN =
READ FILE(RFAMFILE) INTO (RDUMMY ) KEY(RFMAP(RFMAPROW,3));
RFAMREC(1) = RDUMMY;
IF RFMAP(RFMAPROW,4) = 0 THEN DO;
READ FILE(RFAMFILE) INTO (RDUMMY ) KEY (RFMAP(RFMAPROW,4));
RFAMREC(2) = RDUMMY; END;
/* FIXED RECORDS FOR THIS UNIT IS IN RFAMREC(1),RFAMREC(2) */
/* READ THE EXTENTS INTO RFVFILE FOR THIS UNIT */
DO R=5 TO 10; IF RFMAP(RFMAPROW,R)<0 THEN GO TO READY1;
READ FILE(RFAMFILE) INTO(RDUMMY) KEY(RFMAP(RFMAPROW,R));
RFVFILE(R-5) = RDUMMY; END;
READY1:

/*  ****************************************************** PREPROCESSOR SEARCH ******************** */

/* FORMAT: SEARCH ((AN640 <SUBSET I = > AN720 ))
NOTE: WHEN =" IS USED THIS COMMAND ACTS LIKE A TWO WAY SEARCH.
WHEN SUBSET IS USED THIS ACTS AS A ONE WAY SEARCH.
RESTRICTIONS (1) BLANKS MAY APPEAR ANYWHERE EXCEPT WITHIN A KEY.
(2) TWO AND ONLY TWO KEY ANSWERS MAY BE SPECIFIED.
(3) BLANKS MUST SURROUND <SUBSET I = >
(4) THIS COMMAND CANNOT APPEAR WITH A DO LIST COMMAND.
(5) KEY ANSWERS MUST BE LIST ANSWERS NOT FIXED.
*/
% SEARCH: PROCEDURE(TEXT) RETURNS(CHARACTER);
DCL(I,J,REL ) FIXED;
DCL( TEXT, A1, A2, FRONT, TAIL, FINAL ) CHARACTER;
DO I=1 TO 29; IF SUBSTR(TEXT,I,2) = 'AN' THEN GO TO L0; END;
L0: /* I POINTS TO THE 'A' OF 'AN'. */
DO J=I+2 TO 99; IF SUBSTR(TEXT,J,1) = ' ' THEN GO TO L1; END;
L1: A1 = SUBSTR(TEXT,I+2,J-I-2);
DO I=J TO 99; IF SUBSTR(TEXT,I,1) = ' ' THEN DO: REL=0; GO TO L2;
END;
L2: IF SUBSTR(TEXT,I,6) = 'SUBSET' THEN DO: REL=1; GO TO L2; END;
END;
L3: DO J=I+1 TO 99; IF SUBSTR(TEXT,J,2) = 'AN' THEN GO TO L3; END;
L4: A2 = SUBSTR(TEXT,J+2,I-J-2);
MACRO SOURCE2 LISTING

197       FRONT=  CALL SEA(*** || TEXT || ***; || A1 || ' ' || A2)||' ' ||
198       REL || ' ' ||
199       TAIL=  RNST,RSTLIST,TEXT,RFREQ,RFAMREC,RFIXLIMIT,RFDIRCITY,REPRCODE
200       */
201       FINAL=  FRONT || TAIL ;
202       RETURN(FINAL);
203       END SEARCH;
204       */
205       */ ****************************************** PREPROCESSOR CHECK ************** */
206       */ FORMAT: CHECK( <I,D,NUMBER>,<A,E> = <A,E> WITHIN <INTEGER>
207       PERCENT) */
208       RESTRICTIONS: (1) NONE; */
209       /*
210       % CHECK: PROCEDURE(TEXT) RETURNS(CHARACTER);
211       /* THIS PREPROCESSOR PROCEDURE WILL CHECK IF TWO QUANTITIES ARE
212       EQUAL WITHIN A GIVEN PERCENT. */
213       DCL(LEFT,RIGHT,TEXT,ITEM,PERCNT,FINAL,ID)
214       FIXED;
215       DCL( I,J)
216       DO I=1 TO 99; IF SUBSTR(TEXT,I,1)='=' THEN GO TO L1; END;
217       L1:  /* I POINTS TO '=' */
218       DO J=1 TO 50; IF SUBSTR(TEXT,J,1)=',,' THEN GO TO L11; END;
219       L11: /* J POINTS TO THE ',', AFTER THE STATEMENT ID */
220       ID = SUBSTR(TEXT,I+1,J-1-I); I=J;
221       DO J=1 TO 999; IF SUBSTR(TEXT,J,1) = '=' THEN GO TO L2; END;
222       L2:/* J POINTS TO '=' */
223       LEFT= SUBSTR(TEXT,I+1,J-1-I);
224       DO I=J+1 TO 999; IF SUBSTR(TEXT,I,6)='WITHIN' THEN GO TO L3;END;
225       L3:/* I POINTS TO 'WITHIN' */
226       RIGHT= SUBSTR(TEXT,J+1,I-1-J); I=I+6;
227       DO J=1 TO 999; IF SUBSTR(TEXT,J,7)= 'PERCENT' THEN GO TO L4; END;
228       L4:  PERCENT= SUBSTR(TEXT,I,J-1);
229       FINAL= 'RI = '+'|LEFT|'+ ';
230       FINAL= FINAL ||' RJ = '+'|RIGHT|'+ '; RK = '+' |PERCNT|'+ ' ;
231       FINAL= FINAL ||' RN= ABS(RI - RJ); RM=(RK*RK)/100.00; IF RN >
232       RM THEN '+;
233       FINAL= FINAL | '' PUT EDIT(1) CHECK ERROR NO. ('|ID'|' VALUES AND PERCENT
234       T ARE: ''RI, RJ,RK) (SKIP,COL(8),A,(3) (F(5),X(1)))'';
235       RETURN(FINAL);
236       % END CHECK;
237       */
238       */ ****************************************** PREPROCESSOR CKVALUE ************** */
239       */ FORMAT: CKVALUE(1 <I,D,NUMBER>,<A,E> = <A,E> TO <A,E>; ...,
240       <A,E> ; ... <A,E>) */
241       RESTRICTIONS: (1) UP TO 20 <A,E> AND UP TO 10 <A,E> ARE
242       ALLOWED;
243       (2) A '.' MUST TERMINATE THE LIST OF RANGES AND
244       VALUES THAT ARE TO BE CHECKED.
\[(\text{SUBRG, STPG})\]

MACRO SOURCE LISTING

(3) A '; ' MUST SEPARATE THE LIST S OF RANGES AND VALUES.

\[\text{/*}
\]

% CKVALUE: PROCEDURE(TEXT) RETURNS(CHARACTER):

/* THIS PREPROCESSOR PROCEDURE WILL GENERATE CODE TO CHECK IF
THE \(<\text{A,E}>\) ON THE LEFT OF THE EQUAL SIGN EQUALS ANY OF THE
VALUES ON THE RIGHT. IF NOT AN ERROR MESSAGE IS GENERATED */

\text{DCL( TEXT, ITEM, BEGIN, AE, FINAL, ID ) CHARACTER;}
\text{DCL( I, J, X, K, VALUE, RANGE CT ) FIXED;}
\text{DO I=1 TO 99; IF SUBSTR(TEXT, I, 1)='=' THEN GO TO L1; END;
L1: /* I POINTS TO THE BEGINNING OF THE STRING AT '=' */
DO J=I+1 TO 50; IF SUBSTR(TEXT, J, 1)='=' THEN GO TO L11; END;
L11: /* J POINTS TO THE '=' AFTER THE STATEMENT ID */
ID = SUBSTR(TEXT, I+1, J-I-1); I=J;
DO J=I+1 TO 499; IF SUBSTR(TEXT, J, 1)='=' THEN GO TO L2; END;
L2: /* J POINTS TO THE '=' */
AE = SUBSTR(TEXT, I+1, J-I-1); J=J+1; VALUE=0; RANGE CT=0;
BEGIN=''' RI=''' || AE || '=';''
BEGIN LOOP TO ANALYZE THE LIMITS AND VALUES. J POINTS TO
THE BEGINNING OF THE LOOP. VALUECT AND RANGE CT ARE THE COUNTERS
OVER THE VALUES AND RANGES */

LOOP: DO X=1 TO 30; /* 30 IS THE MAXIMUM VALUES AND RANGES ONE CAN
HAVE */

DO I=J TO 999; IF SUBSTR(TEXT, I, 2)='=' THEN GO TO L3;

IF SUBSTR(TEXT, I, 1)='=' THEN GO TO L6;

IF SUBSTR(TEXT, I, 1)='=' THEN GO TO L7; END;

L3: RANGE CT = RANGE CT+1; ITEM = SUBSTR(TEXT, J, I-J);
BEGIN = BEGIN || ' LIMIT(' || RANGE CT || ',1)=' || ITEM || ' ';''
J=I+2; DO I=J TO 999; IF SUBSTR(TEXT, I, 1)='=' THEN GO TO L4;

IF SUBSTR(TEXT, I, 1)='=' THEN GO TO L5; END;

L4: ITEM = SUBSTR(TEXT, J, I-J);
BEGIN = BEGIN || ' LIMIT(' || RANGE CT || ',2)=||ITEM|| ';''
J=I+1; GO TO STOPPL;

L5: ITEM = SUBSTR(TEXT, J, I-J);
BEGIN = BEGIN || ' LIMIT(' || RANGE CT || ',2)=||ITEM|| ';''
BEGIN = BEGIN || ' VALUE(' || VALUE || ')=' || ITEM || ' ';''
J=I+1; GO TO STOPPL;

L7: VALUE = VALUE+1; ITEM = SUBSTR(TEXT, J, I-J);
BEGIN = BEGIN || ' VALUE(' || VALUE || ')=' || ITEM || ' ';''
GO TO OUTLOOP;

STOPPL: END LOOP;

OUTLOOP: /* INPUT STRING HAS BEEN PROCESSED. BELOW IS CODE
TO CHECK IF THE \(<\text{A,E}>\) ON THE LEFT OF THE EQUAL SIGN (RI) MEETS
ANY OF THE REQUIREMENTS OF VALUE OR RLIMIT */
BEGIN = BEGIN || ' RK=' || VALUE || ' RL=' || RANGE CT || ' ';''
FINAL = RM=0; DO RJ=1 TO RK; IF VALUE(RJ)=RI THEN RM=1; END;
DO RJ=1 TO RL; IF RLIMIT(RJ, 1)<=RI & RLIMIT(RJ, 2)>=RI THEN RM=1;
EN D;

IF RM=0 THEN PUT EDIT(***CHECK VALUE ERROR NO. '||ID||***); (SKIP,
MACRO SOURCE2 LISTING

295  COL(8),A; /*
296  FINAL = BEGIN || FINAL;
297  RETURN(FINAL);
298  END CKVALUE;
299  */
300  /* *************** PREPROCESSOR ASSIGN *************** */
301  /* FORMAT: ASSIGN ((AN576 = <A.X.E.>;)) OR ASSIGN ((AN576C = <A.X.E.>;))
302  RESTRICTIONS: (1) ANY NUMBER OF BLANKS MAY SURROUND EACH '*' OR 'u'
303  OR '*' OR 'u'.
304  (2) NO BLANKS MAY BE WITHIN THE KEY ANSWER.
305  (3) THIS COMMAND REQUIRES A '*'.
306  (4) NO MACRO PL/1 COMMANDS CAN FOLLOW AN ASSIGNMENT
307  COMMAND AND AS LITTLE AS POSSIBLE PL/1 CODE
308  SHOULD FOLLOW THE ASSIGNMENTS COMMANDS.
309  (5) THE FIRST ASSIGNMENT COMMAND MUST BE PRECEDEED
310  BY A PREASN COMMAND. */
311  */
312
313  % ASSIGN: PROCEDURE(TEXT) RETURNS(CHARACTER);
314  /* THIS PREPROCESSOR PROCEDURE GENERATES A CALL TO THE RUPDATE
315  PROCEDURE */
316  */
317  DCL(TEXT,ITEM,ANS,SIGN,BACK,FINAL,AE ) CHARACTER;
318  DCL(1,J,RH ) FIXED;
319  DO I=1 TO 29; IF SUBSTR(TEXT,I,2)="AN" THEN GO TO L1; END;
320  L1: I=I+1;
321  /* I POINTS TO *N* OF 'AN' */
322  DO J=I+1 TO 99; ITEM= SUBSTR(TEXT,J,1); IF ITEM="" THEN GO TO L2;
323  THEN GO TO L2;
324  IF ITEM="C" THEN DO; RH=1; GO TO L2; END; END;
325  L2: ANS= SUBSTR(TEXT,I+1,J-1-1);
326  DO J=I TO 99; IF SUBSTR(TEXT,J,1)="*" THEN GO TO L3; END;
327  L3: DO J=I+1 TO 999; IF SUBSTR(TEXT,J,1)="*" THEN GO TO L4; END;
328  L4: AE= SUBSTR(TEXT,I+1,J-1-1);
329  IF RH=1 THEN SIGN= RCH8= 1; ELSE
330  SIGN= RCH8= 1; /* AE INCLUDES THE '*' */
331  RCH8= RCH8; RDIRCMT,RFIXLIM,RFAMREC,RNEST,
332  RSTLIST,TEXT,HEAD, RCH8= 1;
333  RCH8= 1; /* CALL RUPDATE */
334  CALL RUPDATE SIGN || ANS || BACK;
335  RETURN(FINAL);
336  % END ASSIGN;
337
338  /* ******************* PREPROCESSOR WHEN ******************* */
339  /* FORMAT: WHEN((<A.X.E.> <RELATION> <A.X.E. TO A.X.E.>) <<A.X.E.>> ... */
340  /* ... (<A.X.E.> THEN <STATEMENTS>>))
341  RESTRICTIONS: (1) UP TO 20 <A.X.E.> AND 10 <A.X.E. TO A.X.E.> MAXIMUM.
342  (2) ANY MACRO OR PL/1 COMMANDS MAY APPEAR IN THE
343  <STATEMENTS>.*/
MACRO SOURCE2 LISTING

344 (3) A ';' MUST FOLLOW THE '<STATEMENTS>'.
345 (4) BLANK(S) MUST SURROUND THE '<RELATION>'.
346 (5) NO ';' FOLLOWS THE LAST A.E. OR A.E. TO A.E.
347 BEFORE THE 'THEN'.
348 (6) A ';' SEPARATES THE ITEMS IN THE LIST. */
349 % WHEN: PROCEDURE(TXT); DCL ITEM(BEGIN); DCL I,J,X,K,VALUEC,RANGE=".
350 DCL ITEM(BEGIN); DCL I,J,X,K,VALUEC,RANGE=".
351 DCL ITEM(BEGIN); DCL I,J,X,K,VALUEC,RANGE=".
352 DO I=1 TO 99; IF SUBSTR(TXT,I,1)='.' THEN GO TO L1; END;
353 L1: /* I POINTS TO THE BEGINNING OF THE STRING AT '('. */
354 DO J=I+1 TO 499; ITEM=SUBSTR(TXT,J,1);
355 IF ITEM='=' | ITEM='<=' | ITEM='>' | ITEM='->' THEN GO TO L2; END;
356 L2: /* J POINTS TO THE BEGINNING OF THE RELATION */
357 AE=SUBSTR(TXT,I+1,J-I-1); IF SUBSTR(TXT,J+1,1)='.' THEN
358 REL=SUBSTR(TXT,J,1); ELSE REL=SUBSTR(TXT,J,2);
359 J=J+2; VALUEC=0; RANGE=0; BEGIN=' RI=' ||AE||=' '; *
360 /* BEGIN LOOP TO ANALYZE THE LIMITS AND VALUES. J POINTS
361 TO THE BEGINNING OF THE NEXT ITEM TO BE ANALYZED. X IS THE
362 COUNTER OVER THE LOOP. VALUEC AND RANGE ARE THE COUNTERS
363 OVER THE VALUES AND RANGES */
364 LOOP: DO X=1 TO 30; /* 30 IS THE MAX VALUES AND RANGES ONE CAN USE */
365 DO I=J TO 999; IF SUBSTR(TXT,I,2)='TO' THEN GO TO L3;
366 IF SUBSTR(TXT,I,1)='.' THEN GO TO L6;
367 IF SUBSTR(TXT,I,1)='THEN' THEN GO TO L7; END;
368 L3: RANGE= RANGE+1; ITEM=SUBSTR(TXT,J-I-J);
369 BEGIN=BEGIN||'|\LIMIT('||RANGECT||'|\LIMIT('||ITEM||')';
370 J=I+2; DO J=J TO 999;
371 IF SUBSTR(TXT,I,1)='.' THEN GO TO L4;
372 IF SUBSTR(TXT,I,1)='THEN' THEN GO TO L5; END;
373 L4: ITEM=SUBSTR(TXT,J-I-J);
374 BEGIN=BEGIN||'|\LIMIT('||RANGECT||'|\LIMIT('||ITEM||')';
375 J=I+1; GO TO STOPLP;
376 L5: ITEM=SUBSTR(TXT,J-I-J);
377 BEGIN=BEGIN||'|\LIMIT('||RANGECT||'|\LIMIT('||ITEM||')';
378 GO TO OUTLOOP;
379 L6: VALUEC=VALUEC+1; ITEM=SUBSTR(TXT,J-I-J);
380 BEGIN=BEGIN||'|\value('||VALUEC||')'=||ITEM||';';
381 J=I+1; GO TO STOPLP;
382 L7: VALUEC=VALUEC+1; ITEM=SUBSTR(TXT,J-I-J);
383 BEGIN=BEGIN||'|\value('||VALUEC||')'=||ITEM||';';
384 GO TO OUTLOOP;
385 STOPLP: END LOOP;
386 OUTLOOP: /* VALUES AND RANGES HAVE BEEN PROCESSED */
387 /* I POINTS TO THE '}' IN 'THEN' */
388 I=I+4; DO J=I TO 999; IF SUBSTR(TXT,J,1)='.' THEN GO TO L8;
389 END;
390 L8: /* J POINTS TO '}' */
391 STMS=SUBSTR(TXT,J-I-J);
392 BEGIN=BEGIN||'|\Rk='||VALUEC||'; RL='||RANGECT||';';
MACRO SOURCE2 LISTING

393  /* (1) IF THE <RELATION> IS '=' THEN ANY EQUALITY IN RLIMIT OR RVALUES
394    CAUSES <STATEMENTS> TO BE EXECUTED (IMPLIED OR).
395  (2) IF THE RELATION IS '<=' THEN INEQUALITY IN ALL OF RLIMIT AND
396    RVALUES CAUSES <STATEMENTS> TO BE EXECUTED (IMPLIED AND).
397  (3) ANY OTHER RELATION ONLY VALUES ARE CHECKED. IF THE RELATION
398    HOLDS IN ANY ONE OF THEM THEN <STATEMENTS> ARE EXECUTED
399     (IMPLIED OR).
400  */
401  /*(1)*/  IF REL='=' THEN DO;
402    FINAL='RM=O; DO RJ=1 TO RK; IF RI=RVALUE(RJ) THEN RM=1; END;
403    DO RJ=1 TO RL;
404    IF RM=1 THEN DO; '||STMS||' END; ';
405    ELSE
406        /* (2) */ IF REL='=' THEN DO;
407            FINAL='RM=O; DO RJ=1 TO RK; IF RI=RVALUE(RJ) THEN RM=0; END;
408            DO RJ=1 TO RL;
409            IF RI=RLIMIT(RJ,1) & RI <= RLIMIT(RJ,2) THEN RM=0; END;
410            IF RM=1 THEN DO; '||STMS||' END; ';
411    /* (3) */ else
412        FINAL='RM=O; DO RJ=1 TO RK;
413            IF RI=RVALUE(RJ) THEN RM=1; END;
414        IF RM=1 THEN DO; '||STMS||' END; ';
415        FINAL= BEGIN|| FINAL;
416        RETURN(FINAL);
417    % END WHEN;
418    % DEACTIVATE AN;
419  */
420  % DEACTIVATE AN;
421   DCL RCH8 CHAR(8) VARYING;
422   DCL RNEST2 FIXED BIN INIT(0,0);
423   DCL RHHEAD(4000), REXT(4000), PSLIST(4000) FIXED BIN;
424   DCL RERRCODE FIXED BIN;
425   DCL RANSWER FIXED BIN;
426   DCL RAN(4) FIXED BIN INIT(0,0,0,0);
427   DCL RFMAPROW, RHROW1, RHROW2, RHCOL2 FIXED BIN;
428   DCL RFIXLIMIT FIXED BIN INIT(100000);
429   DCL(RI,RJ,RR,RL,RM,RN,RR) FIXED BIN(31,0); 
430   DCL RFAMFILE FILE DIRECT ENV(F7000), REGIONAL(1) KEYED;
431   DCL DRICTRY FILE DIRECT ENV(F34), REGIONAL(1) KEYED;
432   DCL RFAMFILE FILE RECORD SEQUENTIAL INPUT BUFFERED ENV(100000);
433   DCL RHUSEMP FILE RECORD SEQUENTIAL INPUT BUFFERED ENV(100000);
434   DCL RFAMREC2 CHAR(7000);
435   DCL RFVFILE0(0:5) CHAR(7000) EXTERNAL;
436   DCL RFMAP(1200,10) FIXED BIN EXTERNAL;
437   DCL ROUMMY CHAR(7000);
438   DCL RANSØIR CHAR(34);
439   DCL RTYPE CHAR(1) DEF RANSØIR POS(3);
440   DCL RLENGTH CHAR(2) DEF RANSØIR POS(4);
MACRO SOURCE LISTING

442 DCL RSTART CHAR(4) DEF RANSDIR POS(6);
443 DCL RANSIN CHAR(4) DEF RANSDIR POS(10);
444 DCL RMAP(9800,0:3) FIXED BIN EXTERNAL;
445 DCL ROUT FIXED BIN INIT(0);
446 DCL RHMI(16) FIXED BIN INIT(0,2,450,4000,7350,0,2450,4000,7350,0,
447 2400,4900,7350,0,2450,4900,7350);
448 DCL RHM2(16) FIXED BIN INIT(0,0,0,0,1,1,1,2,2,2,3,3,3,3,3,3,3);
449 DCL RFM(1200) FIXED BIN BASED(RFMPTR);
450 DCL RHM(2450) FIXED BIN BASED(RHMPTR);
451 DCL(RVALUE(20),RLIMIT(10,2)) FIXED BIN(31,0);
452 DCL RREDX FIXED BIN(31,0);
453 DCL RDIRMAP(100000) FIXED BIN EXTERNAL;
454 DCL RDIRSPAC FIXED BIN INIT(0);
455 DCL RDINFO(500) CHAR(34) EXTERNAL;
456 DCL UPDATE ENTRY(FIXED DFC,FIXED BIN(31,0), FIXED BIN, FILE
457 DIRECT ENV(F34),REGIONAL(1) KEYED, FIXED BIN, (2) CHAR(7000),
458 (2) FIXED BIN, (4000) FIXED BIN, (4000) FIXED BIN,
459 (4000) FIXED BIN ,FIXED DEC, CHAR(8) VARYING);
460 DCL BREAKS ENTRY(FIXED BIN(31,0),FIXED DEC,FIXED DEC,FIXED DEC,
461 (FIXED DEC ) RETURNS ( FIXED BIN )
462 DCL RSETCMP ENTRY(CHAR(40) VARYING(2,30) FIXED BIN(31,0),
463 FIXED DEC );
464 DCL ANSW ENTRY(FIXED DEC, FILE DIRECT ENV(F34),REGIONAL(1) KEYED,
465 (2) FIXED BIN, (4000) FIXED BIN, (4000) FIXED BIN, (2) CHAR(7000),
466 FIXED BIN ) RETURNS(CHARACTER(20) VARYING);
467 DCL SEA ENTRY( CHAR(40) VARYING(FIXED DEC,FIXED DEC, FIXED DEC, (2) FIXED BIN,(4000) FIXED BIN,(4000) FIXED BIN,
468 (2) CHAR(7000),FIXED BIN FILE DIRECT ENV(F34),REGIONAL(1)
469 KEYED,FIXED BIN );
470 DCL ANSW ENTRY(FIXED DEC, FILE DIRECT ENV(F34),REGIONAL(1) KEYED,
471 (2) FIXED BIN, (4000) FIXED BIN, (4000) FIXED BIN, (2) CHAR(7000),
472 FIXED BIN ) RETURNS(FIXED BIN(31,0));
473 ON CONVERSION BEGIN; ONSOURCE ="O"; END;
474 OPEN FILE (RFAMFILE) UPDATE, FILE(RDIRCTRY) INPUT,
475 FILE(RELFILE) UPDATE, FILE(RDIRCTRY) INPUT;
476 ON KEY(RELFILE) GO TO REOFFAM;
477 ON TRANSMIT(RFAMFILE) BEGIN; PUT EDIT( 'READ ERROR ON RFAMFILE' )
478 (SKIP,A); GO TO RCOLLOP; END;
479 ON TRANSMIT(RDIRCTRY) BEGIN; PUT EDIT( 'READ ERROR ON RDIRCTRY')
480 (SKIP,A); GO TO RREAD; END;
481 ON TRANSMIT(RFAMMAP) BEGIN; PUT EDIT( 'READ ERROR ON RFAMMAP')
482 (SKIP,A); GO TO RREAD; END;
483 (SKIP,A); GO TO RREAD; END;
484 RDIRMAP =0;
485 DO RI=1 TO 10; READ FILE(RFAMMAP) SET (RFMPTR);
486 DO RJ=1 TO 1200; RFMAP(RJ,RI)= RFM(RJ); END; END;
487 DO RI=1 TO 16; READ FILE(RHOUSEMP) SET (RHMPTR);
488 RJ= RHMI(1); AK= RHM2(1); RJ= AK;
489 DO RL=1 TO 2450; RHMAP(RL+RI,RJ) = RHMI(1); END; END;
490 PUT EDIT ( 'FMAP ( FIRST 20 ROWS)' );
(SUBRGN,STPG):

MACRO SOURCE LISTING

491     (RFMAP(RJ,RI) DO RI=1 TO 10 DO RJ=1 TO 20)
492     (SKIP,A,SKIP(2),(20),(SKIP,(10),(10),(F(5),X(1))));
493     PUT SKIP(2);
494     PUT EDIT(RHMAP
495     (FIRST 20 ROWS) *
496     (RFMAP(RJ,RI) DO RJ=0 TO 3 DO RI=1 TO 20))
496     (SKIP,A,SKIP(1),(20),(SKIP,(4),(F(5),X(1))));
497     PUT SKIP;
498     RHMAPROW=0;
499     RHMAPCOL=-1;
500     ROWLOOP: RHMAPROW=RHMAPROW+1; IF RHMAPROW > 9800 THEN GO TO RNORMAL;
501     COLLOOP: RHMAPCOL= RHMAPCOL+1;
502     IF RHMAPCOL >3 THEN DO; RHMAPCOL=-1; GO TO ROWLOOP; END;
503     IF RHMAP(RHMAPROW,RHMAPCOL)<0
504     IF RHMAP(RHMAPROW,RHMAPCOL)=9999 THEN GO TO COLLOOP;
505     /* IN HERE GOES THE I QUIT ROUTINE */
506     RQUIT=RQUIT+1; IF RQUIT>20 THEN DO; PUT EDIT(* I QUIT*)
507     (SKIP,A); GO TO RNORMAL; END;
508     RFMAPROW = RFMAP(RHMAPROW,RHMAPCOL);
509     READ FILE(RFAMFILE) INTO (RDUMMY) KEY(RFMAP(RFMAPROW,3));
510     RFAMREC(1)= RDUMMY;
511     IF RFMAP(RFMAPROW,4) =0 THEN DO;
512     READ FILE(RFAMFILE) INTO (RDUMMY) KEY(RFMAP(RFMAPROW,4));
513     RFAMREC(1)= RDUMMY; END;
514     /* FIXED RECORDS FOR THIS UNIT IS IN RFAMREC(1),RFAMREC(2) */
515     /* READ THE EXTENTS INTO RFVFILE FOR THIS UNIT */
516     DO R=5 TO 10; IF RFMAP(RFMAPROW,R)<0 THEN GO TO READY;
517     READ FILE(RFAMFILE) INTO(RDUMMY) KEY(RFMAP(RFMAPROW,R));
518     RFVFILE(R-5)= RDUMMY; END;
519     READY;
520     PUT EDIT('H=',RHMAPROW,'U=',RHMAPCOL) (SKIP,A(2),F(4),X(1),A(2),
521     F(1));
522     /* THIS UNIT IS READY TO BE EVALUATED. THE FIXED ANSWERS ARE IN RFAMREC. AND THE VARIABLE EXTENTS ARE IN RFVFILE */
523
524     * ACTIVATE AN;
525     /* *************** MACRO PL/1 GOES IN HERE *************** */
526     DCL (1,J,K) FIXED BIN(31,0);
527     DCL CH5 CHARACTER(5);
528     /* ANSWERS 97 TO 110 ARE ELIGIBLE TO BE UPDATED */
529     RANGE((AN97 TO AN110));
530     /* THE SUM OF THE PRODUCTS OF ANSWERS 524 AND 525 IS CHECKED TO
531     BE EQUAL TO THE SUM OF ANSWERS 269 AND 270 WITHIN 1 % */
532     K=0;
533     DC_LIST((ANS269))
534     K= K+ AN((524)) * AN((525));
535     END_LIST
536     CHECK(1, K = AN((269)) * AN((270)) WITHIN 1 PERCENT)
/* CHECK TO MAKE SURE THAT ANSWER 27 IS IN THE RANGE FROM 60 TO 200 */
I = AN((27));
DO_LIST((AN(12)) DO_LIST((AN(149))
IF AN((161)) > 0 THEN DO; CKVALUE((1, I = 60 TO 200.)) END;
END_LIST END_LIST

SEARCH((AN(162) = AN(159))
SEARCH((AN(145) = AN(163))
SEARCH((AN(165) + SUBSET AN(145))

/* CHECK TO SEE IF ANSWER 107 EQUALS 200 OR 22 OR IS IN THE RANGE FROM THE VALUE OF ANSWER 289 TO 100. */
CKVALUE((2, AN((107)) = 200; AN((289)) TO 100; 22))

/* WHEN ANSWER 123 IS IN THE RANGE FROM 0 TO 150 OR FROM 700 TO 20000 THEN A MESSAGE TO THAT EFFECT IS PRINTED */
WHEN(( AN((123)) = 0 TO 150; 700 TO 20000 THEN PUT EDIT(*NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700 *) (SKIP,A));

PREASN
/* ANSWER 110 IS SET EQUAL TO THE SUM OF ANSWERS 107 AND 108. */
J = AN((110)); PUT EDIT(J) (SKIP,F(5));
ASSIGN ((AN(110) = AN((107)) + AN((108))));
J = AN((110)); PUT EDIT(J) (X(3),F(5));

/* ANSWER 97 IS SET EQUAL TO THE BREAK VALUE OF THE SUM OF ANSWERS 107 AND 108. */
ASSIGN((AN(97) = BREAKS ( AN((107)) + AN((108)); 0,1000,50,01));
J = AN((97)); PUT EDIT(J) (X(3),F(2));

ASSIGN((AN(104C) = 'ABCD'));
CH5 = AN((104C)); PUT EDIT(CH5) (X(2),A);

<<<<<<<<<<< END OF MACRO PL/1 >>>>>>>>>> */

% DEACTIVATE AN;

/* ********************************************* PROCEDURE ANSW ********************************************* */

ANSW: PROCEDURE(rDIRCPTY,rRNIGHT,ruREST,ruROST,ruRFAMREC,rRFIXLIMT)
RETURNS(FIXED BIN(31,0));
DCL (LENGTH,HEADER,POINT,START1,LENGTH1,L1,POINT1,EXT1,M1) FIXED BIN;
DCL HEADER1 FIXED BIN;
DCL J FIXED BIN(31,0);
DCL RFAMREC(2) CHAR(7000);
DCL RFVFILE(0+5) CHAR(7000) EXTERNAL;
MACRO SOURCE2 LISTING

DCL I FIXED DECIMAL;
DCL RDIRINFO(500) CHAR(34) EXTERNAL;
DCL RDIR(1000) FIXED BIN EXTERNAL;
DCL (START EXT) FIXED BIN;
DCL RFIXLIMIT, RNEST2(RSTLIST(4000), RREST(4000), K,L,M,N) FIXED BIN;
DCL RDIRTRY FILE DIRECT ENV(F34), REGIONAL(1) KEYED;
DCL RANSDIR CHAR(34);
DCL RTYPE CHAR(1) DEF RANSDIR POS(3);
DCL RLENGTH CHAR(2) DEF RANSDIR POS(4);
DCL RSTART CHAR(4) DEF RANSDIR POS(6);
DCL RANSIN CHAR(4) DEF RANSDIR POS(10);
REDO (( I ))
IF RNEST(1) = 0 & RNEST(2) = 0 THEN GO TO OUTSIDE;
IF RNEST(1) = 0 & RNEST(2) = 0 THEN GO TO INSIDE1;
IF RNEST(1) = 0 & RNEST(2) = 0 THEN GO TO INSIDE2;
ELSE GO TO ERR1;
INSIDE2: IF RNEST(2) = RANSIN THEN GO TO ERR2;
HERE: J = SUBSTR(RFVFILE(REXT(RANSIN)), RSTLIST(RANSIN)+RSTART, RLENGTH);
GO TO RETJ;
INSIDE1: IF RNEST(1) = RANSIN THEN GO TO HERE;
/* WE NOW TREAT THE CASE OF AN IMPLIED SUM OF A SUBLIST Answer */
WHILE INSIDE A SINGLE DO LIST BLOCK */
REDO (( I ))
IF RANSIN = RNEST(1) THEN GO TO ERR3;
START = SUBSTR(RFVFILE(REXT(RANSIN)), RSTLIST(RANSIN)+RSTART, RLENGTH+4, 1);
EXT = SUBSTR(RFVFILE(REXT(RANSIN)), RSTLIST(RANSIN)+RSTART+RLENGTH+4, 1);
L = SUBSTR(RFVFILE(REXT(RANSIN)), RSTLIST(RANSIN)+RSTART, RLENGTH);
REDO (( I ))
J = 0;
DO K = 1 TO L;
J = J + SUBSTR(RFVFILE(EXT), START+RSTART, RLENGTH);
M = SUBSTR(RFVFILE(EXT), START+4, 1);
EXT = SUBSTR(RFVFILE(EXT), START+4, 1);
START = M;
END: GO TO RETJ;
OUTSIDE: IF RANSIN = 0 THEN GO TO OUTSUM;
/* FIX ANSWER */
IF I > RFIXLIMIT THEN K = 2; ELSE K = 1;
J = SUBSTR(RFAMREC(K), RSTART, RLENGTH);
OUTSUM: START = RSTART; LENGTH = RLENGTH; HEADER = RANSIN;
REDO (( I ))
IF RANSIN = 0 THEN GO TO OUTSUMs; IF HEADER > RFIXLIMIT THEN K = 2;
ELSE K = 1;
J = 0;
L = SUBSTR(RFAMREC(K), RSTART, RLENGTH);
POINT = SUBSTR(RFAMREC(K), RSTART+RLENGTH+4, 1);
EXT = SUBSTR(RFAMREC(K), RSTART+RLENGTH+4, 1);
DO M = 1 TO L;
J = J + SUBSTR(RFVFILE(EXT), POINT+START, RLENGTH);
N = SUBSTR(RFVFILE(EXT), POINT+4, 1);
EXT = SUBSTR(RFVFILE(EXT), POINT+4, 1);
MACRO SOURCE2 LISTING

638     POINT= N;     END; GO TO RETJ;
639     OUTSUM= START= RSTART; LENGTH= RLENGTH; HEADER= RANSIN;
640     READ ( RANSIN )
641     IF RANSIN=0, THEN GO TO ERR4:
642     /* WE NOW HAVE A WELL CONSTRUCTED SUBLIST ANSWER */
643     IF HEADER> RFIXLIMIT THEN K=2; ELSE K=1;
644     J=0;  L= SUBSTR(RFAMREC(K),RSTART,RLENGTH);
645     POINT= SUBSTR(RFAMREC(K),RSTART+RLENGTH,4);
646     EXT= SUBSTR(RFAMREC(K),RSTART+RLENGTH+4,1);
647     DO M=1 TO L;
648     L1= SUBSTR(RFVFILE(EXT),START+POIN+LENGTH1,4);  
649     EX1= SUBSTR(RFVFILE(EXT),START+POIN+LENGTH1+4,1);
650     IF L1 <= 0 THEN GO TO WO;
651     DO M1=1 TO L1;
652     J= J+ SUBSTR(RFVFILE(EXT1),POIN1+START+LENGTH1;
653     N= SUBSTR(RFVFILE(EXT1),POIN1,4);
654     EXT1= SUBSTR(RFVFILE(EXT1),POIN1+4,1);
655     POINT1=N;     END;
656     W0;
657     N= SUBSTR(RFVFILE(EXT1),POIN1,4);
658     EXT= SUBSTR(RFVFILE(EXT),POIN1+4,1);
659     POINT= N; END; GO TO RETJ;
660     ERR1:  PUT EDIT('RNEST IS IN ERROR * THIS WAS NOTICED WITHIN AN('',
661     I,')').  RNEST IS: *RNEST(1),RNEST(2) *  
662     ( SKIP,A,F(5),A,F(5),X(1),F(5) );  GO TO ERREXIT;
663     ERR2:  PUT EDIT('ERROR2** AN('',I,') FALLS WITHIN OUTER DO_LIST AN('',
664     RNEST(1),')'), AND INNER DO_LIST AN('',RNEST(2),')'), BUT ANS
665     ' IS CONTAINED WITHIN ANSWER *,RANSIN)
666     ( SKIP,F(5),A,F(5),A,F(5),A,F(5),A,A(4) );  GO TO ERREXIT;
667     ERR3:  PUT EDIT('ERROR3** AN IMPLIED SUM OF SUBLIST ANSWER *,RANSIN)
668     ' HAS OCCURRED WITHIN A DO_LIST AN('',RNEST(1),')') BUT ANS
669     ' IS NOT WITHIN THE SCOPE OF ANSWER *,RNEST(1)
670     ( SKIP,A,F(5),A,F(5),A,F(5),A,F(5) );  GO TO ERREXIT;
671     ERR4:  PUT EDIT('ERROR4** AN IMPLIED SUM OF A SUBLIST ANSWER *,RANSIN)
672     ' OUTSIDE ANY DO_LIST HAS THE CHAIN OF ANSWER CONTAINED WITHIN AS
673     ( SKIP,A,F(5),A,A(4) ) ( F(5) )
674     GO TO ERREXIT;
675     ERREXIT: GO TO RDREAD;
676     RETJ:  RETURN(J);
677     END ANSH;
678     
679     */ ********************************** PROCEDURE RDO ********************************** */
680     RDO:  PROCEDURE ( RANSWER,  RNEST,RSTLIST,REXT,RHEAD,
681     RFAMREC, RFIXLIMIT, RDICTRY, RERRCODE);
682     DCL(RANSWER,RNEST(2),RSTLIST(4000),REXT(4000),RHEAD(4000)) FIXED BIN;
683     DCL RANSDIR CHAR(34);
684     DCL RTYPE CHAR(1) DEF RANSDIR POS(3);
685     DCL RLENGTH CHAR(2) DEF RANSDIR POS(4);
MACRO SOURCE LISTING

687   DCL RSTART CHAR(*4) DEF RANS2DIR POS(6);  
688   DCL RANS2I CHAR(*4) DEF RANS2DIR POS(10);  
689   DCL(R1),RERCODE,RF2XILIMIT ) FIXED BIN;  
690   DCL RDIR2CY DIREC ENV((34),REGIONAL(1)) KEYOU;  
691   DCL RF2VFIL(0:5) CHAR(7000) EXTERNAL;  
692   DCL RF2MREC(2) CHAR(7000);  
693   DCL RDIRINF(500) CHAR(34) EXTERNAL;  
694   DCL RDIRMAP(10000) FIXED BIN EXTERNAL;  
695   RERRCODE=0;  
696   REED ((RANS2WER))  
697   IF RTYPE -= 'H' THEN GO TO RERR_DO;  
698   IF RANS2I=O & RNEST(1)=O THEN GO TO RF2LOOP;  
699   IF RANS2I=O & RANS2I= RNEST(1) THEN GO TO RS2LOOP;  
700   RERR_DO: PUT EDIT(* ERROR IN D2_LIST FOR *,RANS2WER)  
701   (SKIP,A,F(5)); RERRCODE=1; GO TO EN2RD;  
702   RF2LOOP: RNEST(1)= RANS2WER; IF RANS2WER> RF2XILIMIT THEN RI=2;  
703   ELSE RI=1;  
704   RST2ST(RANS2WER)= SUBSTR(RF2MREC(RI),RSTART+RLENGTH,4);  
705   REXT(RANS2WER)= SUBSTR(RF2MREC(RI),RSTART+RLENGTH+4,1);  
706   RHEAD(RANS2WER)= SUBSTR(RF2MREC(RI),RSTART+RLENGTH);  
707   GO TO EN2RD;  
708   RS2LOOP: RNEST(2)=RANS2WER;  
709   RST2ST(RANS2WER)= SUBSTR(RF2VFIL(REXT(RANS2I)),  
710   RST2ST(RANS2I)+ RSTART+RLENGTH,4);  
711   REXT(RANS2WER)= SUBSTR(RF2VFIL(REXT(RANS2I)),  
712   RST2ST(RANS2I)+ RSTART+RLENGTH+4,1);  
713   RHEAD(RANS2WER)= SUBSTR(RF2VFIL(REXT(RANS2I)),  
714   RST2ST(RANS2I)+RSTART+RLENGTH);  
715   EN2RD: RETURN; END RDO;

/* ***************** PROCEDURE BREAKS ******************* */

717   BREAKS: PROCEDURE(AE,LB,UB,INCR,ZERO) RETURNS(FIXED BIN );  
718   /* FORMAT: (1) <VARIBLE>= BREAK(S<A,E>,<INTEGR Suite <INTEGR Suite  
719   <POSITIVE INTEGER>,<0 OR I>);  
720   (2) ASSIGN((A567)= BREAKS(<A,E>,<INTEGR Suite <INTEGR Suite  
721   <POSITIVE INTEGER>,<0 OR I>);  
722   RESTRICTIONS: (1) ONLY<VARIBLE> OR ASSIGN((<KEY ANSWER>) MAY BE USED  
723   ON THE LEFT OF THE "='" SIGN.  
724   (2) THE INTEGR Suite REPRESENT RESPECTIVELY THE LOWER  
725   BOUND, UPPER BOUND, AND INCREMENT. 0 OR I IS  
726   SPECIFIED INDICATING THE TYPE OF BREAK FOR 0.  
727   (3) ONLY ONE SET OF (1) ARE USED.  
728   (4) ** IS REQUIRED AT THE END OF THE STATEMENT.  
729   (5) ANY NUMBER OF BLANKS MAY PRECEDE OF FOLLOW'*(","'),  
730   **,'**BREAKS*,"*/  
731   DCL ( A,E,J ) FIXED BIN (31,0);  
732   DCL ( LB,UB,INCR,ZERO ) FIXED DEC(5,0);  
733   IF ZERO=0 THEN GO TO L2;
(SUBRG, STRG):

MACRO SOURCE2 LISTING

736 L1: /* NO SPECIAL BREAK FOR ZERO */

737 IF AE<LB THEN J=0; ELSE

738 IF AE>UB THEN J=(UB-LB)/INCR+1; ELSE

739 IF MOD((AE-LB), INCR)=0 THEN J=(AE-LB)/INCR; ELSE

740 J=FLOOR((AE-LB)/INCR)+1; GO TO RETJ;

741 L2: /* ZERO GETS A SPECIAL BREAK */

742 IF LB>0 | UB<0 | AE<0 THEN GO TO L1;

743 / * AT THIS POINT LB<=0, UB>=0 AND AE>=0 */

744 IF AE>UB THEN J=(UB-LB)/INCR+2; ELSE IF MOD((AE-LB), INCR)=0

745 THEN J=(AE-LB)/INCR+1; ELSE J=FLOOR((AE-LB)/INCR)+2;

746 RETJ: RETURN(J);

747 END BREAKS;

748

749 /* ****************************************** PROCEDURE SEA ****************************************** */

750 % ACTIVATE AN;

751 SEA: PROCEDURE TEXT,A1,A2,REL, RNEST, RSTLIST, REXT, RHEAD,

752 RFAMREC, RFIXLIMIT, RDIRECTRY, RERRCODE); 

753 DCL TEXT CHAR(40) VARYING;

755 DCL(A1,A2,REL) FIXED DEC;

756 DCL(RANSWER,RNEST(2), RSTLIST(4000), RHEAD(4000), REXT(4000)) FIXED BIN;

757 DCL(RFIXLIMIT, RERRCODE) FIXED BIN;

758 DCL RDIRECTRY FILE DIRECT ENV(F34, REGIONAL(1)) KEYED;

759 DCL RFAMREC(2) CHAR(7000);

760 DCL(I,J,K,L,R) FIXED BIN;

761 DCL RSET(2,30) FIXED BIN (31,0);

762 DCL RANSDIR CHAR(34);

763 DCL RTYPE CHAR(1) DEF RANSDIR POS(31);

764 DCL RANS CHAR(4) DEF RANSCHAR POS(10);

765 DCL RDIRINFO(500) CHAR(34) EXTERNAL;

766 DCL RDIRMAP(10000) FIXED BIN EXTERNAL;

767 IF RNEST(1) =0 | RNEST(2) =0 THEN DO;

768 PUT EDIT(' SEARCH COMMAND OF ANSWERS 'A1', 'A2', 'APPEARS WITHIN A

769 DO_LIST COMMAND. SEARCH NOT PERFORMED ') (SKIP,A,F(5),X(2),F(5),A);

770 GO TO STSEA1; END;

771 RSET = -9999;

772 DO I=1 TO 2;

773 SET1: REED(A1);

774 IF RANSIN=0 THEN DO; PUT EDIT(' ERROR IN SEARCH COMMAND, ONE OR

775 BOTH ANSWERS 'A1', 'A2', 'ARE FIXED ') (SKIP,A,F(5),X(2),F(5),A); GO TO

776 STSEA; END;

777 J = RANSIN;

778 REED(J);

779 IF RANSIN=0 THEN DO; K=0;

780 DO_LIST(ANJ1) K=K+1; RSET(I,K)= AN((A1)); END_LIST END;

781 ELSE DO; K=0; DO_LIST((ANRANSIN)) DO_LIST((ANJ1)) K=K+1;

782 RSET(I,K)= AN((A1)); END_LIST END_LIST END;

783 A1 = A2;

784 END;
MACRO SOURCE2 LISTING

785      /* AT THIS POINT BOTH SETS ARE READY TO BE COMPEARED */
786      CALL RSETCMP( TEXT,RSET,REL1);
787      STSEA:   END SEA;
788
789      % DEACTIVATE AN;
790      /* ******************************************* PROCEDURE ANSWC ******************************************* */
791      ANSWC: PROCEDURE(I ,RDIRCTR, RNEST,RSTLIST,REXT,RFAMREC,
792               RFIXLIMIT)RETURNS(CHARACTER(20) VARYING);
793      DCL I     FIXED DECIMAL (5,0);
794      DCL RNEST(2),RSTLIST(4000),
795          REXT(4000),RFIXLIMIT,K) FIXED BIN;
796      DCL RDIRCTR FILE DIRECT ENV(F34),REGIONAL(1) KEYED;
797      DCL RANSDIR CHAR(34);
798      DCL RTYPE CHAR(1) DEF RANSDIR POS(3);
799      DCL RLEN CHAR(2) DEF RANSDIR POS(4);
800      DCL RSTART CHAR(4) DEF RANSDIR POS(6);
801      DCL RANSIN CHAR(4) DEF RANSDIR POS(10);
802      DCL CH CHAR(20) VARYING;
803      DCL RFAMREC(2) CHAR(7000);
804      DCL RDVFIL(0:5) CHAR(7000) EXTERNAL;
805      DCL RDINFO(500) CHAR(34) EXTERNAL;
806      DCL RDIMAP(10000) FIXED BIN EXTERNAL;
807      DCL REED (( I ))
808      IF RNEST(1)=0 & RNEST(2)=0 THEN GO TO OUTSIDE;
809      ELSE GO TO ERR1;
810      IF RNEST(1)=0 & RNEST(2)=0 THEN GO TO INSIDE1;
811      ELSE GO TO INSIDE2;
812      INSIDE2: IF RNEST(2)=RANS THEN GO TO ERR2;
813      HERE:   CH = SUBSTR(RDVFIL(REXT(RANSIN)),RSTLIST(RANSIN)+RSTART,
814                  RLEN); GO TO RETCH;
815      INSIDE1: IF RNEST(1)=RANS THEN GO TO HERE; ELSE GO TO ERR2;
816      OUTSIDE: IF RANSIN=0 THEN GO TO ERR2;
817      ELSE K=1;
818      CH = SUBSTR(RFAMREC(K),RSTART,RLEN); GO TO RETCH;
819      ERR1:   Put EDIT(*RNEST IS IN ERROR. THIS WAS NOTICED WITHIN AN(*,I,
820                  'C'). RNEST IS: *RNEST(1),RNEST(2))
821      ERR2:   BLK EDIT(* ERROR WITHIN AN(*,I, C). RNEST(1),RNEST(2), AND
822      ANSWER CONTAINED WITHIN FOR*; ARE: *RNEST(1),RNEST(2),RANSIN
823      (SKIP",A,F(5),A,F(5),X(2),F(5) ) ; GO TO ERREXIT;
824      ERREXIT: GO TO RREAD;
825      END ANSWC;
826      /******************************************* PROCEDURE RSETCMP ******************************************* */
827      RSETCMP: PROCEDURE( TEXT,RSET,REL1);
828      DCL TEXT CHAR(40) VARYING;


(SUBRG, STRG):

MACRO SOURCE2 LISTING

834 DCL REL FIXED DEC;
835 DCL(RSET(2,30)) FIXED BIN(31,0);
836 DCL STACK(30) I,J,K FIXED BIN;
837 /* REL=0 IMPLIES '=' REL=1 IMPLIES 'SUBSET' */
838 /* THIS ROUTINE CHECKS THE SETS IN RSET TO BE RELATED ACCORDING TO REL AND ISSUES ERROR MESSAGES IF NOT SATISFIED. ALL OF RSET IS CONSIDERED ELEMENTS TO BE COMPARED UNLESS EQUAL TO -9999 */
840 K=0;
841 STACK=0;
842 DO I=1 TO 30; IF RSET(1,I)=-9999 THEN GO TO HERE;
843 DO J=1 TO 30; IF RSET(2,J)= RSET(1,I) THEN DO;
844 STACK(J)=1; K=1; END; END;
845 IF K=1 THEN DO; K=0; GO TO HERE; END;
846 PUT EDIT(RSET(1,1),') IS IN THE LEFT SIDE BUT NOT IN THE RIGHT SIDE OF THE COMMAND: ' (TEXT) (SKIP,F(6),A,A);
847 HERE: END;
848 IF REL=0 THEN DO;
849 DO I=1 TO 30; IF STACK(I)=0 & RSET(2,I)=-9999 THEN
850 PUT EDIT(RSET(2,1),') IS IN THE RIGHT SIDE BUT NOT LEFT SIDE OF THE COMMAND: ' (TEXT) (SKIP,F(6),A,A);
851 END; END;
852 END RSETCMP;
853 */
854 /* ******************************************** PPREADLIST ******************************************** */
855 RUPDATE: PROCEDURE(I,R,I) RERCCODE, RDIRCTRY, RFIXLIMIT, RFAMREC, RNEST, RSLIST, REXT, RHEAD, CH, RCH8;
856 /* THIS ROUTINE WILL ASSIGN RI TO ANSWER I */
857 RUPDATE: PROCEDURE(I,R,I)
858 DCL I FIXED DECIMAL;
859 DCL CH FIXED DEC;
860 DCL RCH8 CHAR(8) VARYING;
861 DCL RCHVY CHAR(8) VARYING;
862 DCL R(RREST(REX(RSET(2),RSLIST(4000),REXT(4000), RHEAD(4000), J,K,L,M,N) FIXED BIN;
863 DCL RANSOR CHAR(34);
864 DCL RTYPE CHAR(1) DEF RANSOR POS(3);
865 DCL RLENGTH CHAR(2) DEF RANSOR POS(4);
866 DCL RSTART CHAR(4) DEF RANSOR POS(6);
867 DCL RANSIN CHAR(4) DEF RANSOR POS(10);
868 DCL RDIRCTRY FILE DIRECT ENV(F(34),REGIONAL(1)KEYED;
869 DCL RFWFILE(505) CHAR(7000) EXTERNAL;
870 DCL RFAMREC(2) CHAR(7000);
871 DCL RDIREINFO(500) CHAR(34) EXTERNAL;
872 DCL RDIRINAP(10000) FIXED BIN EXTERNAL;
873 RERCCODE=0;
874 /* CHECK IF THE ANSWER IS WITHIN THE LIMITS OF THE RANGE COMMAND */
875 IF I>RAN(1) & I<=RAN(2) THEN GO TO RANGOCR;
876 IF I=RAN(3) & I<=RAN(4) THEN GO TO RANGEOK;
877 PUT EDIT("ERROR*** AN ATTEMPT WAS MADE TO UPDATE ANSWER ", I,' WHICH IS NOT WITHIN THE LIMITS SPECIFIED IN THE RANGE COMMAND.

(SUBRG,STRG);

MACRO SOURCE2 LISTING

883 UPDATE WAS NOT PERFORMED*1 (SKIP,A,F(5),A1; GO TO EXIT;

884 RANGEOK;

885 REED (I 1 I)

886 /* DETERMINE IF WE ARE WITHIN A DO_LIST BLOCK OR NOT */

887 IF RNEST(1)=0 & RNEST(2)=0 THEN GO TO OUTSIDE;

888 IF RNEST(1)>0 & RNEST(2)=0 THEN GO TO INSIDE1;

889 IF RNEST(1)=0 & RNEST(2)>0 THEN GO TO INSIDE2;

890 ELSE GO TO ERR1;

891 OUTSIDE: IF RANSIN==0 THEN GO TO ERR2;

892 IF CH=1 THEN DO; SUBSTR(RFAMREC(K),RSTART,RLENGTH) = RCHVY;

894 GO TO EXIT; END;

895 IF RLENGTH >6 THEN DO;

896 PUT STRING (RCHVY) EDIT(RI) (F(RLENGTH));

897 SUBSTR(RFAMREC(K), RSTART, RLENGTH) = RCHVY; GO TO EXIT;

898 END;

899 PUT STRING (RCHVY) EDIT(RI) (F(RLENGTH));

900 DO N=1 TO RLENGTH ; IF SUBSTR(RCHVY,N,1)='.' THEN

901 SUBSTR(RCHVY,N,1)='0'; END;

902 INSIDE1: IF RNEST(1)== RANSIN THEN DO( L=11 GO TO ERR3; END;

904 HERE1:

905 PUT STRING (RCHVY) EDIT(RI) (F(RLENGTH));

906 SUBSTR(RFAMFILE(REXT(RANSIN)),RSTLIST(RANSIN)+RSTART,

907 RLENGTH) = RCHVY; GO TO EXIT;

908 INSIDE2: IF RNEST(2)== RANSIN THEN DO: L=2; GO TO ERR3; END;

909 GO TO HERE1;

910 ERR1: PUT EDIT('ERROR** RNEST IS IN ERROR, THIS WAS NOTICED WITHIN

911 AN ASSIGN OF 'I,,' RNEST IS: '=',RNEST(1),RNEST(2))

912 (SKIP,A,F(5),A,F(5),X(1),F(5)); GO TO EXIT;

913 ERR2: PUT EDIT('AN ATTEMPT WAS MADE TO UPDATE 'I,,' WHICH IS

914 CONTAINED WITHIN ANSWER 'I,,' WHILE NOT WITHIN ANY DO_LIST

915 BLOCK, UPDATE NOT PERFORMED ') (SKIP,A,F(5),A,F(5),A);

916 GO TO EXIT;

917 ERR3: PUT EDIT('AN ATTEMPT WAS MADE TO UPDATE ANSWER 'I,,' INSIDE

918 A SINGLE DO_LIST BLOCK, 'I,,' IS CONTAINED WITHIN ANSWER 'I,

919 RANSIN,' BUT THE DO_LIST IS OVER HEADER 'I, RNEST(1)'

920 (SKIP,A,F(5),A,F(5),A,F(5),A,F(5)); GO TO EXIT;

921 EXIT: RETURN; END RUPDATE;

922 /* ***********************************************

923 EPILOGUE ****************************************** */

924 GO TO RCOLLBP;

925 REOFFAM: PUT EDIT(' EOF-RFAMFILE, ABNORMAL COMPLETION, ROW ',RFMAPROW,

926 OF RFMAP IS: */ (RFMAP(RFMAPROW,RI) DO RI=1 TO 16))

927 (SKIP,A,F(4),A,F(16),F(4)); GO TO RREAD;

928 RNORMAL: PUT EDIT('NORMAL COMPLETION ***** CONGRADULATIONS*')

929 (SKIP,A);

930 RREAD: END IOWAKE;
(SUBRG,STAG)

NO ERROR OR WARNING CONDITION HAS BEEN DETECTED FOR THIS MACRO PASS.
(SUBRG,STRG):

SOURCE LISTING.

(SUBRG,STRG):
/* IOWKEY - ROGER L. WAINWRIGHT : PROGRAM TO GENERATE IOWA
 KEY VARIABLES USING PL/1 AND MACRO PL/1 */

IOWKEY: PROCEDURE OPTIONS(MAIN);
/* ******************************************** PREPROCESSOR END_LIST *************** */
/* FORMAT: END_LIST */

/* ******************************************** PREPROCESSOR DO_LIST *************** */
/* FORMAT: DO_LIST ((AN547)) OR 
   DO_LIST ((<VARIABLE> = <VARIABLE OR INTEGER> TO AN547)) 
   OR <VARIABLE> OR '=" OR <VARIABLE OR INTEGER>,
   BLANK(S) MUST SURROUND 'TO' AND NO BLANKS MAY APPEAR 
   WITHIN THE KEY ANSWERS. */

/* ******************************************** PREPROCESSOR AN ********************** */
/* FORMAT: AN((647)) OR AN((647C)) 
RESTRICTIONS: (1) C INDICATES THE ANSWER WILL BE RETURNED IN 
   CHARACTER FORM ELSE FIXED BIN(31,0) IS ASSUMED. 
   (2) ANY NUMBER OF BLANKS(ZERO OR MORE) MAY PRECEDE 
   OR FOLLOW 'AN' OR '" OR ')
   (3) THE KEY ANSWER MAY NOT CONTAIN ANY BLANKS. */

/* ******************************************** PREPROCESSOR REED ********************** */

/* ******************************************** PREPROCESSOR RANGE ********************** */
/* FORMAT: RANGE ((AN640 TO AN720)) OR 
   RANGE ((AN640 TO AN720, AN94 TO AN102)) 
RESTRICTIONS: (1) BLANKS MUST SURROUND 'TO'.
   (2) ANY NUMBER OF BLANKS(ZERO OR MORE) MAY FOLLOW 
   'RANGE' OR PRECEDE OF FOLLOW EACH '" OR ')'.
   (3) THERE CAN BE NO BLANKS WITHIN THE KEY ANSWERS. */

/* ******************************************** PREPROCESSOR PRESHN ********************** */

/* ******************************************** PREPROCESSOR SEARCH ********************** */
/* FORMAT: SEARCH ((AN640 <SUBSET | = > AN720 )) 
   NOTE: WHEN '=' IS USED THIS COMMAND ACTS LIKE A TWO WAY SEARCH. */
(SUBRG, STRG):

WHEN SUBSET IS USED THIS ACTS AS A ONE WAY SEARCH.

RESTRICTIONS:
1. BLANKS MAY APPEAR ANYWHERE EXCEPT WITHIN A KEY
   ANSWER OR 'SUBSET'.
2. TWO AND ONLY TWO KEY ANSWERS MAY BE SPECIFIED.
3. BLANKS MUST SURROUND 'SUBSET' = >
4. THIS COMMAND CANNOT APPEAR WITH A DO_LIST COMMAND.
5. KEY ANSWERS MUST BE LIST ANSWERS NOT FIXED.

/* ************************************************************
 * PREPROCESSOR CHECK *************************************************/

/* FORMAT: CHECK({<INumber>,<A.E> = <A.E> WITHIN <INTEGER>
   PERCENT})
RESTRICTIONS: (1) NONE. */

/* ************************************************************
 * PREPROCESSOR CKVALUE *************************************************/

/* FORMAT: CKVALUE({<INumber>,<A.E> = <A.E> TO <A.E>; ...
   <A.E>; ... <A.E>})
RESTRICTIONS: (1) UP TO 20 <A.E> AND UP TO 10 <A.E> TO <A.E> ARE
   ALLOWED.
2. A ';' MUST TERMINATE THE LIST OF RANGES AND
   VALUES THAT ARE TO BE CHECKED.
3. A ';' MUST SEPARATE THE LIST OF RANGES AND VALUES. */

/* ************************************************************
 * PREPROCESSOR ASSIGN *************************************************/

/* FORMAT: ASSIGN ((AN576 = <A.E>; )); OR ASSIGN ((AN576C = <A.E>; ));
RESTRICTIONS: (1) ANY NUMBER OF BLANKS MAY SURROUND EACH 'I' OR 'I'.
2. NO BLANKS MAY BE WITHIN THE KEY ANSWER.
3. THIS COMMAND REQUIRES A 'I';
4. NO MACRO PL/I COMMANDS CAN FOLLOW AN ASSIGNMENT
   COMMAND AND AS LITTLE AS POSSIBLE PL/I CODE
   SHOULD FOLLOW THE ASSIGNMENTS COMMANDS.
5. THE FIRST ASSIGNMENT COMMAND MUST BE PRECEDED
   BY A PREASN COMMAND. */

/* ************************************************************
 * PREPROCESSOR WHEN *************************************************/

/* FORMAT: WHEN({<A.E> <RELATION> <A.E. TO A.E>; <A.E>; ...;
   ... <A.E> THEN <STATEMENTS>})
RESTRICTIONS: (1) UP TO 20 <A.E> AND 10 <A.E. TO A.E.> MAXIMUM.
2. ANY MACRO OR PL/I COMMANDS MAY APPEAR IN THE
   <STATEMENTS>.
(SUBRG,STRG):

(3) A ';' MUST FOLLOW THE <STATEMENTS>.
(4) BLANK(S) MUST SURROUND THE <RELATION>.
(5) NO ';' FOLLOWS THE LAST A.E. OR A.E. TO A.E.
BEFORE THE 'THEN'.
(6) A ';' SEPARATES THE ITEMS IN THE LIST.

/* ****************************************** PROLOGUE ****************************************** */

DCL RCH0 CHAR(8) VARYING;
DCL RNEST(2) FIXED BIN INIT(0,0);
DCL (RHEAD(4000),REXT(4000),RSTLIST(4000)) FIXED BIN;
DCL RERCODE FIXED BIN;
DCL RANSWER FIXED BIN;
DCL RAN(4) FIXED BIN INIT(0,0,0,0);
DCL (RMAPROW(RHROW1,RHROW2,RHCOL2)) FIXED BIN;
DCL RFIXLIMIT FIXED BIN INIT(10000);
DCL(R,RL,RK,RK,RK,RK,RK,RK) FIXED BIN(31,0);
DCL RFAPM FILE DIRECT ENV(F(7000),REGIONAL(1)) KEYED;
DCL RDIRECT FILE DIRECT ENV(F(34),REGIONAL(1)) KEYED;
DCL RFAWMAP FILE RECORD SEQUENTIAL INPUT BUFFERED ENV(CONSECUTIVE);
DCL RHRESP FILE RECORD SEQUENTIAL INPUT BUFFERED ENV(CONSECUTIVE);
DCL RFAWREC(2) CHAR(7000);
DCL RFWFILE(0:5) CHAR(7000) EXTERNAL;
DCL RFWMAP(12000:10) FIXED BIN EXTERNAL;
DCL RDMONLY CHAR(7000);
DCL RDMONLY CHAR(34);
DCL RTYPCH CHAR(1) DEF RANDSRD POS(3);
DCL RLENGTH CHAR(2) DEF RANDSRD POS(4);
DCL RSTART CHAR(4) DEF RANDSRD POS(6);
DCL RANSIN CHAR(4) DEF RANDSRD POS(10);
DCL RHMAP(9000:0:3) FIXED BIN EXTERNAL;
DCL RQOUT FIXED BIN INIT(0);
DCL RHMI(16) FIXED BIN INIT(0,2450,4900,7350,0,2450,4900,7350,0,
2400,4900,7350,0,2450,4900,7350);
DCL RHM(16) FIXED BIN INIT(0,0,0,0,1,1,1,2,2,2,2,3,3,3,3);
DCL RFM(12000) FIXED BIN BASED(RFMTR);
DCL RHM(2450) FIXED BIN BASED(RHMTR);
DCL(RVALUE(20),RLIMIT(10,2)) FIXED BIN(31,0);
DCL RRDIX FIXED BIN(31,0);
DCL RDIRMAP(10000) FIXED BIN EXTERNAL;
DCL RDIRMAP(34) FIXED BIN INIT(0);
DCL RDIRDIX EXTERNAL;
DCL RUPDATE ENTRY(FIXED DEC,FIXED BIN(31,0),FIXED BIN,FILE
DIRECT ENV(F(34),REGIONAL(1)) KEYED,FIXED BIN,(2) CHAR(7000),
(2) FIXED BIN,(4000) FIXED BIN,(4000) FIXED BIN,
(4000) FIXED BIN,FIXED DEC,FIXED DEC,FIXED DEC,
FRENDS (FIXED BIN);
DCL RUPDATE ENTRY(FIXED BIN(31,0),FIXED DEC,FIXED DEC,FIXED DEC,
FRENDS (FIXED BIN);
DCL RSETCMP ENTRY(CHAR(40) VARYING,(2,30) FIXED BIN(31,0),
FIXED DEC);
DCL ANSWC ENTRY(FIXED DEC,FILE DIRECT ENV(F(34),REGIONAL(1)) KEYED,
(2) FIXED BIN ; (4000) FIXED BIN; (4000) FIXED BIN; (2) CHAR(7000); 465
FIXED BIN )
| RETURN;CHARACTER(20) VARYING; 466
| DCL SEALENTY (CHAR(40) VARYING,FIXED DEC,FIXED DEC,FIXED DEC,
(2) FIXED BIN,(4000) FIXED BIN,(4000) FIXED BIN,(4000) FIXED BIN,
(2) CHAR(7000),FIXED BIN, FILE DIRECT ENV(34),REGIONAL(1)) 467
| KEYED,FIXED BIN ); 470
| DCL ANSM ENTRY(FIXED DEC, FILE DIRECT ENV(34),REGIONAL(11) KEYED, 471
| (2) FIXED BIN, (4000) FIXED BIN, (4000) FIXED BIN, (2) CHAR(7000), 472
| FIXED BIN )
| RETURN;FIXED BIN(31,0)); 473
| ON CONVERSION BEGIN: ONSOURCE ="01": END; 474
| OPEN FILE ( RFAMFILE ) UPDATE, FILEI RDI RECT ) INPUT,
| FILE (RFAMMAP) INPUT, FILEI RHOSEMP ) INPUT; 475 1
| ON KEY( RFAMFILE ) GO TO REOFAM; 476
| ON TRANSMIT( RFAMFILE ) BEGIN: PUT EDIT('READ ERROR ON RFAMFILE') 477 1
| (SKIP,A); GO TO RCOLL00PE; END; 479
| ON TRANSMIT( RDI RECT ) BEGIN: PUT EDIT('READ ERROR ON RDDGCTRY') 480 1
| (SKIP,A); GO TO RREAD; END; 481
| ON TRANSMIT( RFAMMAP ) BEGIN: PUT EDIT('READ ERROR ON RFAMMAP') 482
| (SKIP,A); GO TO RREAD; END; 483
| RDIRMAP =0;
| DO RI=1 TO 10; READ FILE( RFAMMAP ) SET (RFMMTR); 484
| DO RJ=1 TO 1200; RFMAP(RJ,RI)=RFM(RJ); END; END; 485
| RHMAP(RJ)=RHM(RJ); 486
| DO RL=1 TO 2450; RHMAP(RL+RJ,RI) = RHMAP(RL); END; END; 487
| PUT EDIT('FMAP (FIRST 20 ROWS)',
| (RFMAP(RI,RI) DO RJ=1 TO 10) DO RI=1 TO 20)); 488
| (SKIP,A,SKIP(2),(20)(SKIP,(10)(F(5),(X(1))))); 489
| PUT SKIP(2); 490
| PUT EDIT('RMAP (FIRST 20 ROWS) ',
| (RHM(RI,RI) DO RJ=0 TO 3) DO RI=1 TO 20)); 491
| (SKIP,A,SKIP(1),(20)(SKIP,(4)(F(5),(X(1)))))); 492
| PUT SKIP; 494
| RHMAPROW=01; 495
| RHMAPCOL=-1; 496
| ROWLOOP: RHMAPROW=RHMAPROW+1; IF RHMAPROW > 9600 THEN GO TO RNAORMAL; 497
| RCOLL00PE: RHMAPCOL=RHMAPCOL+1; 498
| IF RHMAPCOL > 3 THEN DO; RHMAPCOL=-1; GO TO ROWLOOP; END; 499
| IF RHMAP(RHMAPROW,RHMAPCOL)<0 500
| /* IN HERE GOES THE I QUIT ROUTINE */ 501
| (SKIP,A); GO TO RREAD; END; 502
| IF RFMAP(RFMAPROW,4) =0 THEN DO; 503
| /* FIXED RECORDS FOR THIS UNIT IS IN RFAMREC(1),RFAMREC(2) */ 504
| READ FILE( RFAMFILE ) INTO (RDUMMY ) KEY(RFAMAP(RFAMAPROW,3)); 505
| IF RFMAP(RFMAPROW,4) =0 THEN DO; 506
| READ FILE( RFAMFILE ) INTO (RDUMMY ) KEY (RFMAP(RFMAPROW,4)); 507
| RFAMREC(2)=RDUMMY; END; 508
| /* READ THE EXTENTS INTO RFVFILE FOR THIS UNIT */ 509
| DO R=5 TO 10; IF RFMAP(RFMAPROW,R)<0 THEN GO TO READY; 510
READ FILE( RFAMFIL ) INTO( RDUMMY ) KEY( RFMAP( RFMAPROW, R ) );

RFVFIL( R-5 )= RDUMMY; END;

READY;

PUT EDIT( 'H=', RFMAPROW, 'U=', RFMAPCOL ) ( SKIP, A(2), F(4), X(1), A(2),
     F(1) );

/* THIS UNIT IS READY TO BE EVALUATED. THE FIXED ANSWERS ARE IN RFAMREC, AND THE VARIABLE EXTENTS ARE IN RFVFIL */

/* **************** MACRO PL/1 GOES IN HERE *************** */

DCL ( I, J, K ) FIXED BIN(31,0);
DCL C55 CHARACTER(55);
/* ANSWERS 97 TO 110 ARE ELIGIBLE TO BE UPDATED */
RAN(1) = 97; RAN(2) = 110;
/* THE SUM OF THE PRODUCTS OF ANSWERS 524 AND 525 IS CHECKED TO BE EQUAL TO THE SUM OF ANSWERS 269 AND 270 WITHIN 1% */

K = 0;

DECLARE R521 FIXED BIN; RANSWER = 521; CALL RO( RANSWER,
     RNEST, RSTLIST, REXT, RHEAD, RFAMREC, RFIXLIMIT, RDIREF,
     RERCODE ); IF RERCODE = 1 THEN GO TO RDEAD; DO R521 = 1 TO RHEAD(521 )
      K = K + ANSW( 524, RDIREF , RNEST, RSTLIST, REXT, RFAMREC,
     RFIXLIMIT); ANSW( 525, RDIREF , RNEST, RSTLIST, REXT,
     RFAMREC, RFIXLIMIT ); IF RNNEST(1) = RANSWER & RNNEST(2) = RANSWER THEN DO;
     PUT EDIT( 'E' )
       ND_LIST ERROR; RANSWER = 'X'; RANSWER(1) = 'F(6)';
     GO TO RDEAD;

RDEAD; END;

Federal 22(1) = SUBSTR( RFVFIL( REXT( RANSWER )), RSTLIST( RANSWER )+1 );

EXT( RANSWER ) = SUBSTR( RFVFIL( REXT( RANSWER ) ), RSTLIST( RANSWER )+1 );

END; IF RNNEST(1) = RANSWER THEN RNNEST(1) = 0; IF RNNEST(2) = RANSWER THEN DO;

RNNEST(2) = 0; RNNEST = RNNEST(1); END;

RI = K; RJ = 25; ANSW( 269, RDIREF , RNNEST, RSTLIST, REXT,
     RFAMREC, RFIXLIMIT ) + ANSW( 270, RDIREF , RNNEST, RSTLIST, REXT,
     RFAMREC, RFIXLIMIT ); RN = ABS( RI - RJ ); RM = ( RK * RI ) /
100.00; IF RN > 100 THEN PUT EDIT( 'CHECK ERROR NO. 1 VALUES AND PERCENT ARE: ', RI, RJ, RK ) ( SKIP, COL(8), A(1), (3)(F(5),X(1)) );

/* CHECK TO MAKE SURE THAT ANSWER 27 IS IN THE RANGE FROM 60 TO 200 */

I = ANSW( 27, RDIREF , RNNEST, RSTLIST, REXT, RFAMREC, RFIXLIMIT );

DECLARE R12 FIXED BIN; RANSWER = 12; CALL RO( RANSWER,
     RNNEST, RSTLIST, REXT, RFAMREC, RFIXLIMIT, RDIREF,
     RERCODE ); IF RERCODE = 1 THEN GO TO RDEAD; DO R12 = 1 TO RHEAD(12 );

DECLARE R149 FIXED BIN; RANSWER = 149; CALL RO( RANSWER,
     RNNEST, RSTLIST, REXT, RHEAD, RFAMREC, RFIXLIMIT, RDIREF,
     RERCODE ); IF RERCODE = 1 THEN GO TO RDEAD; DO R149 = 1 TO RHEAD(149 );

IF ANSW( 161, RDIREF , RNNEST, RSTLIST, REXT, RFAMREC,
     RFIXLIMIT ) > 0 THEN DO; RI = I; RLIM( 1,1 ) = 60; RLIM( 1,1 )
(SUBRG, STRG1): 1,2) = 200; RK = 0; RL = 1; RM = 0; DO RJ = 1 TO RK; IF RVALUE(RJ) = 1,2) THEN RM = 1; END; DO RJ = 1 TO RK; IF RLIMIT(RJ, 1) = 1,2) THEN RM = 1; END; IF RM = 0 THEN PUT EDIT('CHECK VALUE ERR'); OR NO. 1') (SKIP+, COL(8), A); END; IF RNEST(1) = RANSWER & RNEST(2) = 0 THEN DO; PUT EDIT('E ND_LIST ERROR; RANSWER = ', RANSWER) (SKIP+A, F(6)); GO TO 200; RREAD; END; 1; END; IF RNEST(1) = RANSWER THEN RNEST(1) = 0; IF RNEST(2) = RANSWER THEN DO; RNEST(2) = 0; RANSWER = RNEST(1); END; IF RNEST(1) = 0; CALL SEAI('AN163 = AN159', 163, 159, 0, RNEST, RLIST, REXT, RHEAD, RFAMREC, RFIXLIMIT, RDIRECT, RRERCODE 1); CALL SEAI('AN145 = AN163', 145, 163, 0, RNEST, RLIST, REXT, RHEAD, RFAMREC, RFIXLIMIT, RDIRECT, RRERCODE 1); CALL SEAI('AN163 SUBSET AN145', 163, 145, 1, RNEST, RLIST, REXT, RHEAD, RFAMREC, RFIXLIMIT, RDIRECT, RRERCODE 1); /* CHECK TO SEE IF ANSWER 107 EQUALS 20 OR 22 OR IS IN THE RANGE 289 TO 1000 */ RI = ANSW(107, RDIRECT); RNEST, RLIST, REXT, RFAMREC; RFIXLIMIT) ; RVALUE(1,1) = 200; RLIMIT(1,1) = ANSW(289, 554, 1); RDIRECT, RNEST, RLIST, REXT, RFAMREC, RFIXLIMIT) ; RLIMIT(1,2) = 1,2) = 100; RVALUE(2,2) = 21; RK = 21; RL = 1; RM = 0; DO RJ = 1 TO RK; IF RVALUE(RJ) = RI THEN RM = 1; END; DO RJ = 1 TO RK; IF RLIMIT(RJ, 1) = RI & RLIMIT(RJ, 2) = RI THEN RM = 1; END; IF RM = 0 THEN PUT EDIT('CHECK VALUE ERR NO. 2') (SKIP+, COL(8), A); /* WHEN ANSWER 123 IS IN THE RANGE FROM 0 TO 150 OR FROM 700 TO 2000 THEN A MESSAGE TO THAT EFFECT IS PRINTED */ RI = ANSW(123, RDIRECT); RNEST, RLIST, REXT, RFAMREC, RFIXLIMIT; 2,1) = 700; RLIMIT(2,1) = 150; RLIMIT(2,2) = 20000; RK = 0; RL = 558, 2; RM = 0; DO RJ = 1 TO RK; IF RVALUE(RJ) = RI THEN RM = 1; END; DO RJ = 1 TO RK; IF RLIMIT(RJ, 1) = RI & RLIMIT(RJ, 2) = RI THEN RM = 1; END; IF RM = 558, 2; = 1 THEN DO; PUT EDIT('NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700') (SKIP+A); END; READ RFAMFILE INTO (RDUMMY) KEY(RFMAP(RFMAPROW, 3)); RFAMREC(1) = RDUMMY; IF RFMAP(RFMAPROW, 4) = 0 THEN DO; READ FILE( RFAMFILE ) INTO (RDUMMY) KEY (RFMAP(RFMAPROW, 4)); 561, 2
**SUBRGL,STRG:***

258 RFAMREC(2) = RDUMMY: END;
561 /* FIXED RECORDS FOR THIS UNIT IS IN RFAMREC(1),RFAMREC(2) */
561 /* READ THE EXTENTS INTO RFVFILE FOR THIS UNIT */
561 DO R=5 TO 10: IF RFMAP(RFMAPROW,R)<0 THEN GO TO READY1;
562 READ FILE(RFAMFILE) INTO(RDUMMY) KEY(RFMAP(RFMAPROW,R));
562 RFVFILE(R-5) = RDUMMY: END;
562 READY1:
563 /* ANSWER 110 IS SET EQUAL TO THE SUM OF ANSWERS 107 AND 108. */
563 J = ANSW 110, RDI RECT , RNEST, RST LIST, REXT, RFAMREC,
564 RFIXLIMIT; PUT EDIT(J) (SKIP,F(5)):
565 RI = ANSW 107, RDIRECT , RNEST, RSTLIST, REXT, RFAMREC,
566 RFIXLIMIT + ANSW 108, RDI RECT , RNEST, RSTLIST, REXT, RFAMREC,
567 RFIXLIMIT; CALL RUPDATE(110,RI, RRE RCODE, RDIRECT ,
568 RFAMREC,RFIXLIMIT); CALL RUPDATE(97,RI, RRE RCODE, RDIRECT ,
569 RFIXLIMIT,RFAMREC,RNEST, RSTLIST,REXT,RHEAD, 0,RCH8);
570 J = ANSW 110, RDI RECT , RNEST, RSTLIST, REXT, RFAMREC,
571 RFIXLIMIT; PUT EDIT(J) (X(3),F(5)):
572 /* ANSWER 97 IS SET EQUAL TO THE BREAK VALUE OF THE SUM OF 
573 ANSWERS 107 AND 108. */
573 RI = BREAKS( ANSW 107, RDI RECT , RNEST, RSTLIST, REXT, 
574 RFAMREC,RFIXLIMIT) + ANSW 108, RDI RECT , RNEST, RSTLIST, 
575 REXT,RFAMREC,RFIXLIMIT); CALL RUPDATE(97,RI, 
576 RRE RCODE, RDIRECT , RFIXLIMIT,RFAMREC,RNEST, 
577 RSTLIST,REXT,RHEAD, 0,RCH8);
578 J = ANSW 97, RDI RECT , RNEST, RSTLIST, REXT, RFAMREC,RFIXLIMIT 
579 ); PUT EDIT(J) (X(3),F(2));
580 RCH8 = 'ABCD'; CALL RUPDATE(104,RI, RRE RCODE, RDIRECT , 
581 RFIXLIMIT,RFAMREC,RNEST, RSTLIST,REXT,RHEAD, 1,RCH8);
582 CHS = ANSW( 104, RDI RECT , RNEST, RSTLIST, REXT, RFAMREC,
583 RFIXLIMIT; PUT EDIT(CHS) (X(2),A);
584 */
585 /* *************************************************************************/
586 /*************** PROCEDURE ANSW *********************/
587 280 ANSW: PROCEDURE(1, RDIRECT), RNEST, RSTLIST, REXT, RFAMREC,
588 RFIXLIMIT ; RETURNS(FIXED BIN(31,0)) ;
589 DCL (LENGTH,HEADER,POINT,START1,LENGTH1,L,POINT1,EXT1,M1,FIXED
590 BIN);
591 DCL HEADER1 FIXED BIN;
592 DCL J FIXED BIN(31,0);
593 DCL RFAMREC(2) CHAR(70000);
594 DCL RFVFILE(0:5) CHAR(7000) EXTERNAL;
595 DCL RFVFILE(0:5) CHAR(7000) EXTERNAL;
596 DCL RFAMREC(2) CHAR(70000);
597 DCL RDISRNP(10000) FIXED BIN EXTERNAL;
598 DCL RDISRNP(10000) FIXED BIN EXTERNAL;
599 DCL (START,EXT )FIXED BIN;
600 DCL(RFIXLIMIT, RNEST(2),RSTLIST(4000),REXT(4000),K,L,M) FIXED BIN;
601 DCL (RDIRECT FILE DIRECT ENV(F(34),REGIONAL(1)) KEYED;
DCL RANSIDIR CHAR(34);
DCL TYPE CHAR(1) DEF RANSIDIR POS(31);
DCL RLENGTH CHAR(2) DEF RANSIDIR POS(34);
DCL RSTART CHAR(4) DEF RANSIDIR POS(36);
DCL RANSIDIR CHAR(4) DEF RANSIDIR POS(10);
REEDX = ( I ) ; IF RDIRMAP( REEDX ) = O THEN DO; READ FILE( REEDX = RDIRMAP( REEDX ))) ; EXIT FILE( REEDX = RDIRMAP( REEDX )) ;
RDIRSPAC+1; RDIRINF ( RDIRSPAC ) = RANSIDIR; RDIRMAP ( REEDX ) = RDIRMAP( REEDX ) = END; ELSE RANSIDIR = RDIRINF ( RDIRMAP ( REEDX )) ;
REEDX = RDIRSPAC; EXIT FILE( REEDX = RDIRMAP( REEDX )) ;
IF RNEST(1) = 0 & RNEST(2) = 0 THEN GO TO OUTSIDE;
IF RNEST(1) = 0 & RNEST(2) = 0 THEN GO TO INSIDE1;
IF RNEST(1) = O & RNEST(2) = 0 THEN GO TO INSIDE2;
ELSE GO TO ERR1;
INSIDE2: IF RNEST(2) = RANSIN THEN GO TO ERR2;
HERE: J = SUBSTR(RVFILE(REST(RANSIN)),RSTLIST(RANSIN)+RSTART,RLENGTH);
GO TO RETJ;
INSIDE1: IF RNEST(1) = RANSIN THEN GO TO HERE;
/* WE NOW TREAT THE CASE OF AN IMPLIED SUM OF A SUBLIST ANSWER WHILE INSIDE A SINGLE DO_LIST BLOCK */
REEDX = ( RANSIN ) ; IF RDIRMAP( REEDX ) = O THEN DO; READ FILE( REEDX = RDIRMAP( REEDX ))) ; EXIT FILE( REEDX = RDIRMAP( REEDX )) ;
RDIRSPAC+1; RDIRINF ( RDIRSPAC ) = RANSIDIR; RDIRMAP ( REEDX ) = RDIRMAP( REEDX ) = END; ELSE RANSIDIR = RDIRINF ( RDIRMAP ( REEDX )) ;
REEDX = RDIRSPAC; EXIT FILE( REEDX = RDIRMAP( REEDX )) ;
IF RANSIN = RNEST(1) THEN GO TO ERR3;
START = SUBSTR(RVFILE(REST(RANSIN)),RSTLIST(RANSIN)+RSTART+, RLENGTH+4); 
EXT = SUBSTR(RVFILE(EXT(RANSIN)),RSTLIST(RANSIN)+RSTART,RLENGTH+4); J = SUBSTR(RVFILE(EXT),START+4,1); L = SUBSTR(RVFILE(EXT),START+4,1); START = M = END; GO TO RETJ;
OUTSIDE: IF RANSIN = 0 THEN GO TO OUTSUM;
/* FIX ANSWER */
IF > RFILES LIMIT THEN K = 2 ELSE K = 1; 
J = SUBSTR(RVFILE(EXT),START+RSTART,RLENGTH);
M = SUBSTR(RVFILE(EXT),START+4,1); 
EXT = SUBSTR(RVFILE(EXT),START+4,1); 
START = M = END; GO TO RETJ;
OUTSUM: START = RSTART; LENGTH = RLENGTH; HEADER = RANSIN;
REEDX = ( RANSIN ) ; IF RDIRMAP( REEDX ) = O THEN DO; READ FILE( REEDX = RDIRMAP( REEDX ))) ; EXIT FILE( REEDX = RDIRMAP( REEDX )) ;
RDIRSPAC+1; RDIRINF ( RDIRSPAC ) = RANSIDIR; RDIRMAP ( REEDX ) = RDIRMAP( REEDX ) = END; ELSE RANSIDIR = RDIRINF ( RDIRMAP ( REEDX )) ;
REEDX = RDIRSPAC; EXIT FILE( REEDX = RDIRMAP( REEDX )) ;
IF RANSIN=0 THEN GO TO OUTSUM; IF HEADER=RFIXLIMIT THEN K=2; J=0; L = SUBSTR(RFAMREC(K), RSTART, RLENGTH); OUTSUM = SUBSTR(RFAMREC(K), RSTART+RLENGTH, 4); IF RFIXMAP( REEDX, POINT+4,1) THEN DO: READ FILE( RDIRMAP ) INTO(RANSIND) KEY( REEDX ); RDIRSPACE++; RDIRINF( RDIRSPACE)= RANSIND; RDIRMAP( REEDX, POINT+4,1) = RDIRSPACE; END; ELSE RANSIND = RDIRINF( RDIRMAP( REEDX, POINT+4,1) ); IF RANSIN=0 THEN GO TO ERR4; /* WE NOW HAVE A WELL CONSTRUCTED SUBLIST ANSWER */ IF HEADER1=RFIXLIMIT THEN K=2; ELSE K=1; J=0; L = SUBSTR(RFAMREC(K), RSTART, RLENGTH); OUTSUM = SUBSTR(RFAMREC(K), RSTART+RLENGTH, 4); IF RFIXMAP( REEDX, POINT+4,1) THEN DO: READ FILE( RDIRMAP ) INTO(RANSIND) KEY( REEDX ); RDIRSPACE++; RDIRINF( RDIRSPACE)= RANSIND; RDIRMAP( REEDX, POINT+4,1) = RDIRSPACE; END; ELSE RANSIND = RDIRINF( RDIRMAP( REEDX, POINT+4,1) ); IF L <= 0 THEN GO TO WOW; DO M1 = 1 TO L; J = J + SUBSTR(RFAMFILE(EXT1), POINT+START, RLENGTH); END; N = SUBSTR(RFAMFILE(EXT1), POINT+4,1); POINT = N; END; WOW: EXT = SUBSTR(RFAMFILE(EXT, POINT+4,1); IF ERR1=0 THEN GO TO RETJ; ERR1: PUT EDIT( 'RNEST IS IN ERROR, THIS WAS NOTICED WITHIN AN( , I, ') ), RNEST IS: , RNEST1(1), RNEST2(2) ; GO TO ERR2EXIT; ERR2: PUT EDIT( 'ERROR2** AN( ,I,') ) FALLS WITHIN OUTER DO_LIST AN( , RNEST1(1), ) AND INNER DO_LIST AN( , RNEST2(2), ), BUT ANSWER RNEST1, I; ERR3: PUT EDIT( 'ERROR3** AN IMPLIED SUM OF SUBLIST ANSWER', I, 'HAS OCCURRED WITHIN A DO_LIST AN( , RNEST1(1), ) BUT ANSWER RNEST1, I, IS NOT WITHIN THE SCOPE OF ANSWER RNEST1, I' ); GO TO ERR2EXIT; ERR4: PUT EDIT( 'ERROR4** AN IMPLIED SUM OF A SUBLIST ANSWER', I, 'OUTSIDE ANY DO_LIST HAS THE CHAIN OF ANSWER CONTAINED WITHIN AS FOLLOWS: ', HEADER, HEADER1, RANSIN ), SKIP, A, F5, A, A(4), F5, I; GO TO ERR2EXIT; ERR2EXIT: GO TO RREAD; RETJ: RETURN(I);
/* **************************** PROCEDURE RDO **************************** */

RDO:  PROCEDURE (RANSWER, RNEST, RSTLIST, REXIT, RHEAD, 
        RFAMREC, RFIXLIMIT, RDIRECT, RERRCODE); 
       DCL (RANSWER, RNEST, RSTLIST(4000), REXIT(4000), RHEAD(4000)) FIXED BIN; 
       DCL RANSDIR CHAR(34); 
       DCL RTYPE CHAR(1) DEF RANSDIR POS(3); 
       DCL RLENGTH CHAR(2) DEF RANSDIR POS(4); 
       DCL RSTART CHAR(4) DEF RANSDIR POS(6); 
       DCL RANSIN CHAR(4) DEF RANSDIR POS(10); 
       DCL(RI, RERRCODE, RFIXLIMIT) FIXED BIN; 
       DCL RDIRECT FILE DIRECT ENV(F(34), REGIONAL(1)) KEYED; 
       DCL RFVFILE(105) CHAR(7000) EXTERNAL; 
       DCL RFAMREC(2) CHAR(7000); 
       DCL RDIRINF(500) CHAR(34) EXTERNAL; 
       DCL RDIRMAP(10000) FIXED BIN EXTERNAL; 
       DCL(RDirmap, REEDX, RDIRSPAC, REEDX) = 0 THEN DO; READ FILE 
       ( RDIRECT ) INTO(RANSDIR) KEY ( REEDX ) ; RDIRSPAC = 
       RDIRSPAC+1; 
       RDIRINF(RDIRSPAC) = RANSDIR; 
       RDIRMAP(REEDX) = RDIRMAP(REEDX) 
       ELSE RANSDIR = RDIRINF(RDIRMAP(REEDX)) 
       IF RTYPE = 'H' THEN GO TO RERR_DO; 
       IF RANSIN = 0 & RNEST(1) = 0 THEN GO TO RFLOOP; 
       RERR_DO:  PUT EDIT( 'ERROR IN RDo_LIST FOR 'RANSWER) 
       (SKIP, A, F(5)); RERRCODE = 1; GO TO ENDDO; 
       RFLOOP:  RNEST(1) = RANSWER; 
       IF RANSWER > RFIXLIMIT THEN RI = 2; 
       ELSE RI = 1; 
       RSTLIST(RANSWER) = SUBSTR(RFAMREC(RI),RSTART+RLENGTH+4); 
       REXIT(RANSWER) = SUBSTR(RFAMREC(RI),RSTART+RLENGTH+4,1); 
       RHEAD(RANSWER) = SUBSTR(RFAMREC(RI),RSTART+RLENGTH); 
       GO TO ENDDO; 
       RFLOOP:  RNEST(2) = RANSWER; 
       RSTLIST(RANSWER) = SUBSTR(RFVFILE(REXT(RANSIN)), 
       RSTLIST(RANSIN)+RSTART+RLENGTH+4); 
       REXIT(RANSWER) = SUBSTR(RFVFILE(REXT(RANSIN)), 
       RSTLIST(RANSIN)+RSTART+RLENGTH+4,1); 
       RHEAD(RANSWER) = SUBSTR(RFVFILE(REXT(RANSIN)), 
       RSTLIST(RANSIN)+RSTART+RLENGTH); 
       GO TO ENDDO; 
       ENDDO:  RETURN; END RDO; 

/* **************************** PROCEDURE BREAKS **************************** */

BREAKS: PROCEDURE(AE, LB, UB, INCR, ZERO) RETURNS(FIXED BIN); 
/* FORMAT: (1) <VARIABLE> = BREAKS(<A.E.>, <INTEGER>, <INTEGER>) , 
    <POSITIVE INTEGER>, <POSITIVE INTEGER>; (2) ASSIGN((A567)) = BREAKS(<A.E.>, <INTEGER>, <INTEGER>) , 
    <POSITIVE INTEGER>, <POSITIVE INTEGER>; */

RESTRICTIONS: (1) ONLY <VARIABLE> OR ASSIGN((<KEY ANSWER>)) MAY BE USED
(SUBRG, STRG):

ON THE LEFT OF THE "*" SIGN:

(2) THE INTEGERS REPRESENT RESPECTIVELY THE LOWER
ROUND, UPPER BOUND AND INCREMENT. 0 OR 1 IS
SPECIFIED INDICATING THE TYPE OF BREAK FOR 0.
(3) ONLY ONE SET OF (1) ARE USED.
(4) "*" IS REQUIRED AT THE END OF THE STATEMENT.
(5) ANY NUMBER OF BLANKS MAY PRECEDE OF FOLLOW(",","",BREACKs
*/

DCL ( AE, J ) FIXED BIN (31, 0);
DCL ( LB, UB, INCR, ZERO ) FIXED DEC (5, 0);
IF ZERO=0 THEN GC TO L2;
L1: /* NO SPECIAL BREAK FOR ZERO */
    IF AE=0 THEN J=0; ELSE
    IF AE> LB THEN J=(UB-LB)/INCR+1; ELSE
    IF MOD((AE-LB), INCR) = 0 THEN J=(AE-LB)/INCR; ELSE
    J=FLOOR((AE-LB)/INCR)+1; GO TO RETJ;
    L2: /* ZERO GETS A SPECIAL BREAK */
    IF LB>0 OR UB>0 AND AE=0 */
    IF AE> LB THEN J=(UB-LB)/INCR+2; ELSE IF MOD((AE-LB), INCR) = 0
    THEN J=(AE-LB)/INCR+1; ELSE J=FLOOR((AE-LB)/INCR)+2;
    RETJ: RETURN (J);
END BREAKS;

/*/ ****************************************** PROCEDURE SEA *************** */

SEA: PROCEDURE (TEXT, A1, A2, REL, RNAME, RSTLIST, REXT, RHEAD,
RFAMREC, RFLXLIMIT, RDIRECT, RERRCODE);
DCL TEXT CHAR (40) VARYING;
DCL (A1, A2, REL ) FIXED DEC;
DCL (RANSIR, RNEST(2), RSTLIST (4000), RHEAD (4000), REXT(4000)) FIXED BIN;
DCL (RFLXLIMIT, RERRCODE) FIXED BIN;
DCL RDIRECT FILE DIRECT ENV (F34), REGIONAL (1) KEYED;
DCL RFAMREC (2) CHAR (7000);
DCL (I, J, K, L, RJ ) FIXED BIN;
DCL RSET (2, 30) FIXED BIN (31, 0);
DCL RANSIR CHAR (34);
DCL RTEYP CHAR (1) DEF RANSIR POS (3);
DCL RANSIN CHAR (4) DEF RANSIR POS (10);
DCL RDIRINF (500) CHAR (34), EXTERNAL;
DCL RDIRMAP (10000) FIXED BIN EXTERNAL;
IF RNEST(1) = 0 OR RNEST(2) = 0 THEN DO;
PUT EDIT(' SEARCH COMMAND OF ANSWERS ' , A1, A2, ' APPEAR WITHIN A
DD_LIST COMMAND, SEARCH NOT PERFORMED ' ) ( SKIP, A, F (5), X (2), F (5), A );
GO TO STSEA; END;
RSET = -9999;
DO I=1 TO 2;
SET1: REEDX = (A1) ; IF RDIRMAP ( REEDX ) = 0 THEN DO; READ FILE(adj
RDIRECT ); INTO ( RANSIR ) KEY ( REEDX ) ; RDIRSPAC =
RDIRSPAC + 1;
FAIL: RDIRINF ( RDIRSPAC ) = RANSIR; RDIRMAP ( REEDX ) = 773 2
(SUBRGN, STPG):
PROCEDURE ANSWC

DCL I FIXED DECIMAL (5,0);
DCL ( RNEST(2), RSTLIST(4000) ),
REXT(4000), RFIXLIMIT, K FIXED BIN;
DCL RTYPE CHAR(1) DEF RANSIP POS(4);
DCL RLENGTH CHAR(2) DEF RANSIP POS(4);
DCL RSTART CHAR(4) DEF RANSIP POS(6);
DCL RANSIN CHAR(4) DEF RANSIP POS(10);
DCL CH CHAR(20) VARYING;
DCL RFAMREC(2) CHAR(7000);
DCL RFFVFILE(0:5) CHAR(7000) EXTERNAL;
DCL RDIRINF(500) CHAR(34) EXTERNAL;
DCL RDIRMAP(10000) FIXED BIN EXTERNAL;

READX = ( I ) ;
IF RDIMAP( READX ) = 0 THEN DO;
READ FILE( RDIMAP( READX )
) INTO( RANSIP ) KEY( READX ) ;
REEDX = ( I ) ;
RDIFSPAC+1;
RDIRINF( RDIFSPAC ) = RANSIP ;
RDIMAP( READX ) = RDIRINF( RDIMAP( READX )
REEDX ) ;
IF RNEST(1) = 0 & RNEST(2) = 0 THEN GO TO OUTSIDE;
IF RNEST(1) = 0 & RNEST(2) = 0 THEN GO TO INSIDE1;
ELSE GO TO ERR1;
INSIDE1: IF RNEST(1) = RANSIN THEN GO TO ERR2;
HERE: CH = SUBSTR( RFFVFILE( REXT( RANSIN ) ), RSTLIST( RANSIN ) ) + RSTART,
RLNGTH ;
GO TO RETCH;
INSIDE2: IF RNEST(2) = RANSIN THEN GO TO ERR2;
OUTSIDE: IF RANSIN = 0 THEN GO TO ERR2;
IF CH = SUBSTR( RFAMREC(1), RSTART, RLENGTH ) THEN GO TO RETCH;
ERR1: PUT EDIT('RNEST IS IN ERROR. THIS WAS NOTED WITHIN AN("", I, 'C")', RNEST IS: 'RNEST(1), RNEST(2)'
(SKIP,A,F(5),A,F(5),X(2),F(5) ) ;
GO TO ERREXIT;
ERR2: PUT EDIT('ERROR WITHIN AN("",I, 'C")', RNEST(1), RNEST(2), AND
ANSWER CONTAINED WITHIN FOR:
(SKIP,A,F(5),A,F(5),A,F(5),X(2),F(5),X(2),A(4) );
GO TO ERREXIT;

PROCEDURE RSETCMP

DCL TEXT CHAR(40) VARYING;
DCL REL FIXED DEC;
DCL( RSET(2), 50 ) FIXED BIN(31,0);
DCL STACK(30),I,J,K FIXED BIN;
/* REL=0 IMPLIES ' = ' . REL=1 IMPLIES ' SUBSET ' */
/* THIS ROUTINE CHECKS THE SETS IN RSET TO BE RELATED ACCORDING
TO REL AND ISSUES ERROR MESSAGES IF NOT SATISFIED. ALL OF
RSET IS CONSIDERED ELEMENTS TO BE COMPARED UNLESS EQUAL TO -9999 */

K=0;
STACK=0;
DO I=1 TO 30; IF RSET(1,I)=-9999 THEN GO TO HERE;
DO J=1 TO 30; IF PSET(2,J)=RSET(1,I) THEN DO;
STACK(J)=1; K=1; END;
END;
IF K=1 THEN DO; K=0; GO TO HERE; END;
PUT EDIT RSET(1,I), ' IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: ','TEXT) (SKIP,F(6),A,A);
HERE: END;
IF REL=0 THEN DO;
DO I=1 TO 30; IF I=0 & RSET(2,I)=-9999 THEN
PUT EDIT RSET(2,I), ' IS IN THE RIGHT SIDE BUT NOT LEFT SIDE OF
THE COMMAND: ','TEXT) (SKIP,F(6),A,A);
END;
END;
/* **************************************** PPROCEDURE RUPDATE ******************************************* */
RUPDATE: PROCEDURE RI, RERRCODE, RDIRECT, RFIXLIMIT, RFAMREC, RREST, RSETLIST, REXT, RHEAD, RCH8, RCH8V;
/* THIS ROUTINE WILL ASSIGN RI TO ANSWER I */
DCL RI FIXED DECIMAL;
DCL I FIXED DECIMAL;
DCL CH FIXED DEC;
DCL RCH8 CHAR(8) VARYING;
DCL RCH8V CHAR(8) VARYING;
DCL(RERRCODE, RFIXLIMIT, RREST, RSETLIST(4000), REXT(4000), RHEAD(4000), J,K,L,M,N) FIXED BIN;
DCL RANSDIR CHAR(34);
DCL RTYPE CHAR(1) DEF RANSDIR POS(3);
DCL RLENGTH CHAR(2) DEF RANSDIR POS(4);
DCL RRSTART CHAR(4) DEF RANSDIR POS(6);
DCL RRANS CHAR(4) DEF RANSDIR POS(10);
DCL RRDIIC FILE DIRECT ENV(F(34),REGIONAL(1)) KEYED;
DCL RFVFILE(0:5) CHAR(7000) EXTERNAL;
DCL RFAMREC(2) CHAR(7000);
DCL RRDIRINF(500) CHAR(34) EXTERNAL;
DCL RRDIRMAP(10000) FIXED BIN EXTERNAL;
RERRCODE=0;
/* CHECK IF THE ANSWER IS WITHIN THE LIMITS OF THE RANGE COMMAND*/
IF I>=RAN(1) & I <= RAN(2) THEN GO TO RANGEOK;
IF I>= RAN(3) & I <= RAN(4) THEN GO TO RANGEOK;
PUT EDIT 'ERROR*** AN ATTEMPT WAS MADE TO UPDATE ANSWER ',
I,' WHICH IS NOT WITHIN THE LIMITS SPECIFIED IN THE RANGE COMMAND;
UPDATE WAS NOT PERFORMED' (SKIP,A,F(5),A); GO TO EXIT;
(SUBRG, STRG):

732  RANGEOK:
733  REEDX = ( I ) : IF RDIRMAP( REEX ) = 0 THEN DO: READ FILE( 884
734  RDIRECT ) INTO(RANDIR1 KEY ( REEDX ) ) ; RDIRSPAC = 885 2
736  RDIRSPAC+1: RDIRINF( RDIRSPAC) = RANDIR1; RDIRMAP ( REEDX 885 2
737  ) = RDIRSPAC; END; ELSE RANDIR1 = RDIRINF( RDIRMAP ( 885 2
738  REEDX ) ) ;
739  /* DETERMINE IF WE ARE WITHIN A DO_LIST BLOCK OR NOT */
740  IF RNEST(1)=0 & RNEST(2)=0 THEN GO TO OUTSIDE;
741  IF RNEST(1)=0 & RNEST(2)=0 THEN GO TO INSIDE1;
743  IF RNEST(1)=0 & RNEST(2)=0 THEN GO TO INSIDE2;
745  ELSE GO TO ERR1;
746  OUTSIDE: IF RANSIN==0 THEN GO TO ERR2;
748  IF 1> RFIXLIMIT THEN K=2: ELSE K=1;
750  IF CH=1 THEN DO: SUBSTR(RFMREC(K),RSTART,RLENGTH) = RCHB;
752  GO TO EXIT; END;
754  IF RLENGTH > 6 THEN DO:
756  PUT STRING (RCHVY) EDIT(RI) (F(RLENGTH));
758  SUBSTR(RFMREC(K),RSTART,RLENGTH) = RCHVY; GO TO EXIT;
760  END;
762  PUT STRING (RCHVY) EDIT(RI) (F(RLENGTH));
764  DO N=1 TO RLENGTH ; IF SUBSTR(RCHVY,N,1)=' ' THEN
766  SUBSTR(RCHVY,N,1)='0';
768  END;
770  SUBSTR(RFMREC(K),RSTART,RLENGTH) = RCHVY; GO TO EXIT;
772  INSIDE1: IF RNEST(1)== RANSIN THEN DO: L=1; GO TO ERR3; END;
774  HERE1:
776  PUT STRING (RCHVY) EDIT(RI) (F(RLENGTH));
778  SUBSTR(RFAMELE(RNEST(RANSIN)),RSTLIST(RANSIN)+RSTART,
780  RLENGTH) = RCHVY;GO TO EXIT;
782  INSIDE2: IF RNEST(2)== RANSIN THEN DO: L=2; GO TO ERR3; END;
784  GO TO HERE1;
786  ERR1: PUT EDIT('ERROR** RNEST IS IN ERROR, THIS WAS NOTICED WITHIN 900
788  AN ASSIGN OF ' 'I',' RNEST IS: 'RNEST(1),RNEST(2)')
790  SKIP,A,F(5),A,F(5),X(1),F(5) ;
792  GO TO EXIT;
794  ERR2: PUT EDIT('AN ATTEMPT WAS MADE TO UPDATE ',I,' WHICH IS 912
796  CONTAINED WITHIN ANSWER ', RANSIN, ' WHILE NOT WITHIN ANY DO_LIST 913
798  BLOCK, UPDATE NOT PERFORM 914
799  ED ')
801  SKIP,A,F(5),A,F(5),A;
803  GO TO EXIT;
805  ERR3: PUT EDIT(' AN ATTEMPT WAS MADE TO UPDATE ANSWER ',I,' INSIDE 915
807  A SINGLE DO_LIST BLOCK ')
809  'I,' IS CONTAINED WITHIN ANSWER ', 916
811  RANSIN, ' BUT THE DO_LIST IS OVER HEADER ', RNEST(L) 917
815  GO TO EXIT;
817  EXIT; RETURN; END UPDATE;
819  /* *************************************************************/
821  EPILOGUE 822
823  GO TO PCOLLUP;
825  REOFFAM: PUT EDIT(' EOF-RFAMFILE, ABNORMAL COMPLETION, ROW ',RFMAPROW,
827  ' OF RFMAP IS ', (RFMAP(RFMAPROW,RI) DO RI=1 TO 16.)
829  )
831  (SKIP,A,F(6),A,(16)F(4));
833  GO TO RREAD;
(SUBRG, STRG):

796  RNORMAL:  PUT EDIT(*NORMAL COMPLETION ***** CONGRADULATIONS*)

797  (SKIP,A):

928  RREAD:  END IOWAKEY;
### FMAP (FIRST 20 ROWS)

<table>
<thead>
<tr>
<th>1160</th>
<th>2319</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>-1</td>
<td>3</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>4</td>
<td>-1</td>
<td>5</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>6</td>
<td>-1</td>
<td>7</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>8</td>
<td>-1</td>
<td>9</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>10</td>
<td>-1</td>
<td>11</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>12</td>
<td>-1</td>
<td>13</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>14</td>
<td>-1</td>
<td>15</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>16</td>
<td>-1</td>
<td>17</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>18</td>
<td>-1</td>
<td>19</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>20</td>
<td>-1</td>
<td>21</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>22</td>
<td>-1</td>
<td>23</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>31</td>
<td>1</td>
<td>24</td>
<td>-1</td>
<td>25</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>33</td>
<td>1</td>
<td>26</td>
<td>-1</td>
<td>27</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>28</td>
<td>-1</td>
<td>29</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>36</td>
<td>1</td>
<td>30</td>
<td>-1</td>
<td>31</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>38</td>
<td>1</td>
<td>32</td>
<td>-1</td>
<td>33</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>39</td>
<td>1</td>
<td>34</td>
<td>-1</td>
<td>35</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>44</td>
<td>1</td>
<td>36</td>
<td>-1</td>
<td>37</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

### HMAP (FIRST 20 ROWS)

<table>
<thead>
<tr>
<th>9999</th>
<th>9999</th>
<th>9999</th>
<th>9999</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>9999</td>
<td>9999</td>
<td>9999</td>
</tr>
<tr>
<td>3</td>
<td>9999</td>
<td>9999</td>
<td>9999</td>
</tr>
<tr>
<td>9999</td>
<td>9999</td>
<td>9999</td>
<td>9999</td>
</tr>
<tr>
<td>9999</td>
<td>9999</td>
<td>9999</td>
<td>9999</td>
</tr>
<tr>
<td>622</td>
<td>937</td>
<td>9999</td>
<td>9999</td>
</tr>
<tr>
<td>622</td>
<td>9999</td>
<td>9999</td>
<td>9999</td>
</tr>
<tr>
<td>623</td>
<td>9999</td>
<td>9999</td>
<td>9999</td>
</tr>
<tr>
<td>9999</td>
<td>9999</td>
<td>9999</td>
<td>9999</td>
</tr>
<tr>
<td>624</td>
<td>793</td>
<td>9999</td>
<td>9999</td>
</tr>
<tr>
<td>625</td>
<td>794</td>
<td>9999</td>
<td>9999</td>
</tr>
<tr>
<td>626</td>
<td>795</td>
<td>9999</td>
<td>9999</td>
</tr>
<tr>
<td>9999</td>
<td>9999</td>
<td>9999</td>
<td>9999</td>
</tr>
<tr>
<td>9999</td>
<td>9999</td>
<td>9999</td>
<td>9999</td>
</tr>
<tr>
<td>7</td>
<td>9999</td>
<td>9999</td>
<td>9999</td>
</tr>
</tbody>
</table>

### Notes:
- **H=**
  - 2 UnO
  - 7 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
  - 0 IS IN THE RIGHT SIDE BUT NOT LEFT SIDE OF
  - 0 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
  - 1 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
  - 4 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
  - 4 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
  - 3 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
  - 7 IS IN THE RIGHT SIDE BUT NOT LEFT SIDE OF
  - 7 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
- SIDE OF THE COMMAND: (AN163 = AN159)
- THE COMMAND: (AN163 = AN159)
- SIDE OF THE COMMAND: (AN145 = AN163)
- SIDE OF THE COMMAND: (AN145 = AN163)
- SIDE OF THE COMMAND: (AN145 = AN163)
- SIDE OF THE COMMAND: (AN145 = AN163)
- SIDE OF THE COMMAND: (AN145 = AN163)
- SIDE OF THE COMMAND: (AN145 = AN163)
- THE COMMAND: (AN145 = AN163)
- SIDE OF THE COMMAND: (AN163 SUBSET AN145)
- CHECK VALUE ERROR NO. 2
NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700
0 55 3 ABCDE
H = 3 U = 0
1 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
CHECK VALUE ERROR NO. 2
NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700
0 0 1 ABCDE
H = 6 U = 0
CHECK ERROR NO. 1 VALUES AND PERCENT ARE: 00000 17760 1
1 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
3 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
3 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
CHECK VALUE ERROR NO. 2
NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700
900 50 2 ABCDE
H = 6 U = 1
NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700
0 0 1 ABCDE
H = 8 U = 0
CHECK ERROR NO. 1 VALUES AND PERCENT ARE: 0 2971 1
1 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
3 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
3 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
CHECK VALUE ERROR NO. 2
NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700
0 70 3 ABCDE
H = 8 U = 1
NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700
0 0 1 ABCDE
H = 9 U = 0
CHECK ERROR NO. 1 VALUES AND PERCENT ARE: 49800 112 1
11 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
12 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
CHECK VALUE ERROR NO. 2
NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700
0 0 1 ABCDE
H = 11 U = 0
CHECK ERROR NO. 1 VALUES AND PERCENT ARE: 0 300 1
1 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
11 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
CHECK VALUE ERROR NO. 2
NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700
0 0 1 ABCDE
H = 13 U = 0
CHECK ERROR NO. 1 VALUES AND PERCENT ARE: 88000 9760 1
1 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
4 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
4 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
3 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
CHECK VALUE ERROR NO. 2
NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700
240 200 5 ABCDE
H = 13 U = 1
1 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
4 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
4 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
3 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT
SIDE OF THE COMMAND: (AN145 = AN163)
NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700

H= 14 U=0
CHECK ERROR NO. 1 VALUES AND PERCENT ARE: 0 725 1
CHECK VALUE ERROR NO. 2

NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700

H= 16 U=0
CHECK ERROR NO. 1 VALUES AND PERCENT ARE: 10000 4300 1
CHECK VALUE ERROR NO. 2

NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700

H= 16 U=1
CHECK VALUE ERROR NO. 1

NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700

H= 17 U=0
21 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT SIDE OF THE COMMAND: (AN145 = AN163)
CHECK VALUE ERROR NO. 2

NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700

H= 17 U=1
21 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT SIDE OF THE COMMAND: (AN145 = AN163)
CHECK VALUE ERROR NO. 2

NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700

H= 20 U=0
CHECK ERROR NO. 1 VALUES AND PERCENT ARE: 0 611 1
1 IS IN THE RIGHT SIDE BUT NOT LEFT SIDE OF THE COMMAND: (AN163 = AN159)
4 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT SIDE OF THE COMMAND: (AN145 = AN163)
4 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT SIDE OF THE COMMAND: (AN145 = AN163)
4 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT SIDE OF THE COMMAND: (AN145 = AN163)
3 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT SIDE OF THE COMMAND: (AN145 = AN163)
CHECK VALUE ERROR NO. 2

NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700

H= 21 U=0
CHECK ERROR NO. 1 VALUES AND PERCENT ARE: 0 25 1
1 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT SIDE OF THE COMMAND: (AN145 = AN163)
CHECK VALUE ERROR NO. 2

NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700

H= 22 U=0
CHECK ERROR NO. 1 VALUES AND PERCENT ARE: 80000 2765 1
1 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT SIDE OF THE COMMAND: (AN145 = AN163)
CHECK VALUE ERROR NO. 2

NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700

H= 23 U=0
21 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT SIDE OF THE COMMAND: (AN145 = AN163)
CHECK VALUE ERROR NO. 2
NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700
H = 23 U = 1
21 IS IN THE LEFT SIDE BUT NOT IN THE RIGHT SIDE OF THE COMMAND: (AN145 = AN163)
CHECK VALUE ERROR NO. 2
NUMBER OF ACRES IS LESS THAN 150 OR GREATER THAN 700
0 0 1 ABCDE
I QUIT
NORMAL COMPLETION ***** CONGRADULATIONS