EDITING BY COMPUTER

APPENDICES C & D
PROCEDURE OPTIONS (MAIN):

* IMPLEMENTS FRED PRE-PROCESSOR;
* DECLARE (ITEM,CUSTOMER,UNITQTY,DICRATE,INSRATE) FIXED DECIMAL (5,0),
  (UNITVAL,DICOUNT,INSURANCE,FREIGHT,_ITEM CODE FIXED BIN:
  ON ENDFILE (SYSIN) GO TO ENDING;
* READ CARD AND EDIT IT #
  SYSDRIT (CARO) (A80);
  IT (CUSTMXER,ITEM,UNITQTY,UNITVAL,COST,
  INSURANCE,FREIGHT,BILLED) FIXED DECIMAL (7,2), FRTRATE FIXED DECIMAL (5,2),
  CALL INVOICE;
* PUT FILE (SYSPRINT) SKIP(60);
* READ: GET FILE /
  GET STRING(CARD) ED
* CLOSE LISTINGS /
* INVOICE:PROCEDURE
* DECLARE Q #D1AG(60) PACKED CHARACTER (80),
  ITEM (Q #ITEM), Q #CUSTOMER, Q #UNITQTY, Q #UNITVAL, Q #DISCRADE, Q #INSRATE,
  Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  IT (EM CODE OUT OF RANGE
  READ : GET FILE= (Q #D1GCT), FIXED BINARY;
  READ: Q #UNITQTY, Q #UNITVAL, Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
  Q #DISCOUNT, Q #DISCOUNT, Q #INSURANCE, Q #FREIGHT,
LEVEL 2JAN67

OS/360 PL/I COMPILER (F)

PL/I F COMPILER OPTIONS SPECIFIED ARE AS FOLLOWS--
S, A, ST, SIZE=108000, SM=(001,080)

THE COMPLETE LIST OF OPTIONS USED DURING THIS COMPIlATION IS--

   • EACDIC
   • CARGO
   • NOMACRO
   • SOURCE2
   • COMP
   • SOURCE
   • ATR
   • NOXREF
   • NOXREF
   • NOLIST
   • LOAD
   • NODECK
   • FLAG
   • STMT
   • SIZE=108000
   • LINESIZE=60
   • DPT=00
   • SORMGIN=(001,080)
<table>
<thead>
<tr>
<th>CUSTOMEER</th>
<th>****</th>
<th>CUSTOMEER</th>
<th>UNITVAL</th>
<th>COST</th>
<th>FREIGHT</th>
<th>COST</th>
<th>BILLED</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISCRATE</td>
<td>****</td>
<td>DISCRATE</td>
<td>DISCOUNT</td>
<td>INSURANCE</td>
<td>FREIGHT</td>
<td>COST</td>
<td>BILLED</td>
</tr>
<tr>
<td>INSURANCE</td>
<td>****</td>
<td>INSURANCE</td>
<td>INSURANCE</td>
<td>INSURANCE</td>
<td>FREIGHT</td>
<td>COST</td>
<td>BILLED</td>
</tr>
<tr>
<td>FREIGHT</td>
<td>****</td>
<td>FREIGHT</td>
<td>INSURANCE</td>
<td>FREIGHT</td>
<td>COST</td>
<td>BILLED</td>
<td></td>
</tr>
<tr>
<td>COST</td>
<td>****</td>
<td>UNITVAL</td>
<td>UNITVAL</td>
<td>UNITVAL</td>
<td>INSURANCE</td>
<td>FREIGHT</td>
<td>COST</td>
</tr>
<tr>
<td>BILLED</td>
<td>****</td>
<td>BILLED</td>
<td>BILLED</td>
<td>BILLED</td>
<td>BILLED</td>
<td>BILLED</td>
<td>BILLED</td>
</tr>
<tr>
<td>FRTRATE</td>
<td>****</td>
<td>FRTRATE</td>
<td>FRTRATE</td>
<td>FRTRATE</td>
<td>BILLED</td>
<td>BILLED</td>
<td>BILLED</td>
</tr>
</tbody>
</table>
END OF THIS COMPILATION, NO ERRORS WERE NOTED IN THE EDITING STATEMENTS
ERROR 'COST INVALID FOR QUANTITY'

/* CHECK DISCOUNT RATE AND DISCOUNT */
IF DISCRATE=0,1,5,12,20 THEN
  IF DISCOUNT => ALLOW FOR ROUNING */
C <=(COST*DISCRATE/100)-.02,
C >=(COST*DISCRATE/100)+.02 THEN
  ERROR 'DISCOUNT MISCALCULATED'
ELSE
  **
ELSE
  ERROR 'DISCOUNT RATE INVALID'

/* CHECK INSURANCE RATE AND INSURANCE */
IF INSURATE=0,2,3,5,7 THEN
  IF INSURANCE => ALLOW FOR ROUNING */
C <=(COST*INSURATE/100)-.02,
C >=(COST*INSURATE/100)+.02 THEN
  ERROR 'INSURANCE MISCALCULATED'
ELSE
  **
ELSE
  ERROR 'INSURANCE RATE INVALID'

/* CHECK FREIGHT, COMPUTE % FREIGHTAGE */
IF FREIGHTED THEN
  COST/FREIGHT*10 THEN
    QUERY 'FREIGHT HIGH'
  ELSE
    IF COST=0 THEN
      DEFINE FTRATE=0
    ELSE
      DEFINE FTRATE=FREIGHT/COST
  END EDIT INVOICE
RESULTS FOR 29 ACCEPTED RECORDS ARE

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVERAGE MAXIMUM TEMP</td>
<td>75.290</td>
</tr>
<tr>
<td>AVERAGE MINIMUM TEMP</td>
<td>47.500</td>
</tr>
<tr>
<td>TOTAL RAINFALL</td>
<td>9.220</td>
</tr>
<tr>
<td>AVERAGE HOURS SUNSHINE</td>
<td>9.060</td>
</tr>
</tbody>
</table>
DECLARE (ITEM,CUSTOMER,UNITQTY,DISC,INSRATE) FIXED DECIMAL (5,0),
CARD CHARACTER (80),
(UNITVAL,DISCOUNT,INSURANCE,FREIGHT,COST,BILLED) FIXED DECIMAL (7,2), FRTRATE FIXED DECIMAL (5,2), ERRCODE FIXED BIN;

ON ENDFILE (SYSIN) GO TO ENDING;
PUT FILE (SYSPRINT) SKIP(60);
READ:
GET FILE (SYSIN) EDIT (CARD) (A(80));
GET STRING (CARD) EDIT (CUSTOMER,ITEM,UNITQTY,UNITVAL,COST,
DISC,DISCOUNT,INSRATE,INSURANCE,FREIGHT,BILLED)
((2 F(5,0),F(4,0),2 F(6,2),F(2,0),F(6,2),F(2,0),3 F(6,2));
CALL INVOICE;
GO TO READ;
END: CLOSE LISTINGS */
PUT FILE (SYSPRINT) EDIT (1 END OF LISTING ***) (SKIP(60),
A(20));
START EDIT INVOICE
/* CODED AS INTERNAL PROCEDURE */
EDIT (ITEM,CUSTOMER,UNITQTY,DISC,INSRATE) FIXED DECIMAL (5,0),
(UNITVAL,DISCOUNT,INSURANCE,FREIGHT,COST,BILLED) FIXED DECIMAL
(7,2), FRTRATE FIXED DECIMAL (5,2)
LIST CUSTOMER,ITEM,UNITQTY,UNITVAL,COST,DISC,DISCOUNT,
INSRATE,INSURANCE,FREIGHT,FRTRATE,BILLED
HEADING '*** INVOICE EDIT RUN ***'
END OF DECLARATIVE STATEMENTS */
/* FIRST CHECK ITEM CODE AND CUSTOMER CODE FOR VALIDITY */
IF ITEM=0,23127 THEN
*** THE DECLARATIVE SECTION MUST CONTAIN ONE EDIT, ONE FLAG AND ONE LIST OR PRINT ONE STATEMENT
ERROR 'ITEM CODE OUT OF RANGE'
ELSE
**
IF CUSTOMER=0,36381 THEN
ERROR 'CUSTOMER CODE OUT OF RANGE'
ELSE
**
/* CHECK BILLED AMOUNT = COMPONENT SUM */
IF BILLED=COST+INSURANCE+FREIGHT-DISC
ELSE
/**
/* TRY CALCULATED DISCOUNT OR INSURANCE */
C ABS(BILLED-COST-INSURANCE-FREIGHT)*DISC/100)<.02 THEN
ADJUST DISC=COST+INSURANCE+FREIGHT-BILLED,
*** ADJUSTMENT IS TO INVALID ITEM
C "DISCOUNT RECALCULATED",
ABS(BILLED-COST-INSURANCE+FREIGHT+DISCOUNT)<.02 THEN
ADJUST INSURANCE=BILLED-COST+DISCOUNT-FREIGHT,
C "INSURANCE RECALCULATED"
ELSE
ERROR 'BILLED COST DOES NOT AGREE'
/* CHECK COST */
IF COST=UNITQTY*UNITVAL THEN
**
FRED PRE-PROCESSOR

**ERROR 'COST INVALID FOR QUANTITY'

/* CHECK DISCOUNT RATE AND DISCOUNT */
IF DISCRATE=0,=5,=12,=20 THEN
  IF DISCOUNT /* ALLOW FOR ROUNING */
  C <=(COST*DISCRATE/100)-.02,
  C >=(COST*DISCRATE/100)+.02 THEN
    ERROR 'DISCOUNT MISCALCULATED*
  ELSE
    "DISCOUNT RATE INVALID"
ELSE**

/* CHECK INSURANCE RATE AND INSURANCE */
IF INSRATE=0,=2,=3,=5,=7 THEN
  IF INSURANCE /* ALLOW FOR ROUNING */
  C <=(COST*INSRATE/100)-.02,
  C >=(COST*INSRATE/100)+.02 THEN
    ERROR 'INSURANCE MISCALCULATED*
  ELSE
    "DISCOUNT RATE INVALID"
ELSE**

/* CHECK FREIGHT, COMPUTE % FREIGHTAGE */
IF FREIGHT=0 THEN
  **
  IF COST/FREIGHT<=10 THEN
    QUERY "FREIGHT HIGH"
  ELSE
    "FREIGHT TIME"
  ELSE
    IF COST=0 THEN
      DEFINE FRTRATE=0
      ** LHS OF Assignment ILLEGAL
    ELSE
      DEFINE FRTRATE=FREIGHT/COST
  END EDIT INVOICE
 END
<table>
<thead>
<tr>
<th>ITEM</th>
<th>CUSTOMER</th>
<th>UNITQTY</th>
<th>UNITVAL</th>
<th>COST</th>
<th>DISCRATE</th>
<th>DISCOUNT</th>
<th>INSURANCE</th>
<th>FREIGHT</th>
<th>COST</th>
<th>BILLED</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITEM</td>
<td>CUSTOMER</td>
<td>UNITQTY</td>
<td>UNITVAL</td>
<td>COST</td>
<td>DISCRATE</td>
<td>DISCOUNT</td>
<td>INSURANCE</td>
<td>FREIGHT</td>
<td>COST</td>
<td>BILLED</td>
</tr>
<tr>
<td>UNITVAL</td>
<td>UNITQTY</td>
<td>UNITVAL</td>
<td>UNITVAL</td>
<td>COST</td>
<td>DISCRATE</td>
<td>DISCOUNT</td>
<td>INSURANCE</td>
<td>FREIGHT</td>
<td>COST</td>
<td>BILLED</td>
</tr>
<tr>
<td>DISCOUNT</td>
<td>DISCRATE</td>
<td>INSRATE</td>
<td>DISCOUNT</td>
<td>INSURANCE</td>
<td>FREIGHT</td>
<td>COST</td>
<td>BILLED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISCOUNT</td>
<td>DISCRATE</td>
<td>INSRATE</td>
<td>DISCOUNT</td>
<td>INSURANCE</td>
<td>FREIGHT</td>
<td>COST</td>
<td>BILLED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISCOUNT</td>
<td>DISCRATE</td>
<td>INSRATE</td>
<td>DISCOUNT</td>
<td>INSURANCE</td>
<td>FREIGHT</td>
<td>COST</td>
<td>BILLED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FREIGHT</td>
<td>DISCRATE</td>
<td>INSRATE</td>
<td>DISCOUNT</td>
<td>INSURANCE</td>
<td>FREIGHT</td>
<td>COST</td>
<td>BILLED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COST</td>
<td>DISCRATE</td>
<td>INSRATE</td>
<td>DISCOUNT</td>
<td>INSURANCE</td>
<td>FREIGHT</td>
<td>COST</td>
<td>BILLED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BILLED</td>
<td>DISCRATE</td>
<td>INSRATE</td>
<td>DISCOUNT</td>
<td>INSURANCE</td>
<td>FREIGHT</td>
<td>COST</td>
<td>BILLED</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FREIGHT</td>
<td>DISCRATE</td>
<td>INSRATE</td>
<td>DISCOUNT</td>
<td>INSURANCE</td>
<td>FREIGHT</td>
<td>COST</td>
<td>BILLED</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FRED PRE-PROCESSOR ***
END BILLER;

END OF THIS COMPIATION, ERRORS WERE NOTED AS SIGNALLED ABOVE.
DO NOT ATTEMPT TO RUN THIS PROGRAM UNTIL THESE ERRORS HAVE BEEN REMOVED.
**FRED PRE-PROCESSOR***

**BILLER:**
PROCEDURE OPTIONS(MAIN); /* SAMPLE EDITION NO 2, TO DEMONSTRATE FRED PRE-PROCESSOR.*/

**DECLARE**
ITEM,CUSTOMER,UNITOTY,UNITIVAL,CREDIT,INSURANCE,FREIGHT,COST,BILLED, FIXED
DIGITAL (4,0),
UNITIVAL,DISCOUNT,INSURANCE,FREIGHT,COST,BILLED, FIXED
DIGITAL (5,0),
UNITIVAL,DIGITAL (5,0),
FRATED FIXED DECIMAL (5,2), ERRCODE FIXED BIN;
ON END FILE(SYSIN) GO TO ENDING;
PUT FILE(SYSPRINT) SKIP(60);
/* READ CARD AND EDIT IT */
READ:
GET FILE(SYSIN) EDIT (CARD) A(80); /* GET STRING(CARD) EDIT (CUSTOMER,ITEM,UNITOTY,UNITIVAL,CREDIT, 
DISCOUNT,INSURANCE,FREIGHT,COST,BILLED) */
(2 F(5,3),F(5,0),2 F(6,2),F(12,2),F(2,0),3 F(6,2));
CALL INVOICE;
/* CLOSE LISTINGS */
ENDING: PUT FILE(SYSPRINT) EDIT (" END OF LISTING ***") (SKIP(60),
A(80));
START EDIT INVOICE
/* CODED AS INTERNAL PROCEDURE */
EDIT (ITEM,CUSTOMER,UNITOTY,UNITIVAL,CREDIT,INSURANCE,FREIGHT,COST,BILLED, FIXED
DIGITAL (5,0),
UNITIVAL,DIGITAL (5,0),
FRATED FIXED DECIMAL (5,2),
LIST CUSTOMER,ITEM,UNITOTY,UNITIVAL,CREDIT,DISCOUNT,
CREDIT,INSURANCE,FREIGHT,FRATED,BILLED
/* END OF DECLARATIVE STATEMENTS */
/* FIRST CHECK ITEM CODE AND CUSTOMER CODE FOR VALIDITY */
IF ITEM=0,3127 THEN
ERROR 'ITEM CODE OUT OF RANGE'
ELSE **
IF CUSTOMER=0,6381 THEN
ERROR 'CUSTOMER CODE OUT OF RANGE'
ELSE **
IF BILLED=COST+INSURANCE+FREIGHT-DISCOUNT THEN
** ELSE /* TRY CALCULATED DISCOUNT OR INSURANCE */
C ABS(BILLED-COST-INSURANCE-FREIGHT+DISCOUNT)/100<.02 THEN
ADJUST DISCOUNT=COST+INSURANCE+FREIGHT-BILLED;
C 'DISCOUNT RECALCULATED'
ABS(BILLED-COST-CREDIT)/100-FREIGHT+DISCOUNT)<.02 THEN
ADJUST INSURANCE=BILLED-COST+DISCOUNT-FREIGHT;
C 'INSURANCE RECALCULATED'
ELSE
ERROR 'BILLING COST DOES NOT AGREE'
/* CHECK BILLED AMOUNT = COMPONENT SUM */
IF BILLED=COST+INSURANCE+FREIGHT-DISCOUNT THEN
** ELSE /* TRY CALCULATED DISCOUNT OR INSURANCE */
C ABS(BILLED-COST-CREDIT)/100-FREIGHT+DISCOUNT)<.02 THEN
ADJUST DISCOUNT=COST+INSURANCE+FREIGHT-BILLED;
C 'DISCOUNT RECALCULATED'
ABS(BILLED-COST-CREDIT)/100-FREIGHT+DISCOUNT)<.02 THEN
ADJUST INSURANCE=BILLED-COST+DISCOUNT-FREIGHT;
C 'INSURANCE RECALCULATED'
ELSE
ERROR 'BILLING COST DOES NOT AGREE'
/* CHECK COST */
IF COST=UNITOTY*UNITIVAL THEN
** ELSE
WEATHER: PROCEDURE OPTIONS (MAIN):

<table>
<thead>
<tr>
<th>DCL NO.</th>
<th>IDENTIFIER</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>219</td>
<td>Q_#002</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>365</td>
<td>Q_#003</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>22</td>
<td>Q_#ADJUST</td>
<td>AUTOMATIC, BINARY, FIXED(15,0)</td>
</tr>
<tr>
<td>416</td>
<td>Q_#COMPLT</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>23</td>
<td>Q_#DAY</td>
<td>AUTOMATIC, DECIMAL, FIXED(3,0)</td>
</tr>
<tr>
<td>22</td>
<td>Q_#DIAG</td>
<td>(*) , AUTOMATIC, PACKED, STRING, CHARACTER</td>
</tr>
<tr>
<td>22</td>
<td>Q_#DIGCT</td>
<td>AUTOMATIC, BINARY, FIXED(15,0)</td>
</tr>
<tr>
<td>23</td>
<td>Q_#FALL</td>
<td>AUTOMATIC, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>22</td>
<td>Q_#FATAL</td>
<td>AUTOMATIC, BINARY, FIXED(1,0)</td>
</tr>
<tr>
<td>24</td>
<td>Q_#FORM</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>23</td>
<td>Q_#HRS</td>
<td>AUTOMATIC, DECIMAL, FIXED(3,0)</td>
</tr>
<tr>
<td>136</td>
<td>Q_#L001</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>203</td>
<td>Q_#L002</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>282</td>
<td>Q_#L003</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>349</td>
<td>Q_#L004</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>23</td>
<td>Q_#MAX</td>
<td>AUTOMATIC, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>23</td>
<td>Q_#MIN</td>
<td>AUTOMATIC, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>23</td>
<td>Q_#MONTH</td>
<td>AUTOMATIC, STRING, CHARACTER</td>
</tr>
<tr>
<td>22</td>
<td>Q_#QUERY</td>
<td>AUTOMATIC, BINARY, FIXED(1,0)</td>
</tr>
<tr>
<td>6</td>
<td>READCARD</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td></td>
<td>SYSIN</td>
<td>FILE, EXTERNAL</td>
</tr>
<tr>
<td></td>
<td>SYSPRINT</td>
<td>FILE, EXTERNAL</td>
</tr>
<tr>
<td>2</td>
<td>TALLY</td>
<td>AUTOMATIC, INITIAL, DECIMAL, FIXED(3,0)</td>
</tr>
<tr>
<td>61</td>
<td>TST2</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>1</td>
<td>WEATHER</td>
<td>ENTRY, DECIMAL, FLOAT(SINGLE)</td>
</tr>
</tbody>
</table>
WEATHER: PROCEDURE OPTIONS (MAIN):

COMPILER-DIAGNOSTIC MESSAGES

THE SUPPRESSION OF WARNING MESSAGES HAS BEEN REQUESTED.

TIME FOR THIS COMPILATION WAS 1.48 MINUTES
<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Value</th>
<th>Error Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2/3</td>
<td></td>
<td>61.10</td>
<td>RECORD IN ERROR</td>
<td>HOURS TOO HIGH</td>
</tr>
<tr>
<td>2/5</td>
<td></td>
<td>54.02</td>
<td>RECORD IN ERROR</td>
<td>DAY IN Valid</td>
</tr>
<tr>
<td>2/7</td>
<td></td>
<td>68.13</td>
<td>RECORD IN ERROR</td>
<td>MIN &gt; MAX</td>
</tr>
<tr>
<td>2/9</td>
<td></td>
<td>68.13</td>
<td>RECORD IN ERROR</td>
<td>MONTH INVALID</td>
</tr>
<tr>
<td>2/11</td>
<td></td>
<td>57.37</td>
<td>RECORD IN ERROR</td>
<td>DAY IN Valid</td>
</tr>
<tr>
<td>2/14</td>
<td></td>
<td>30.01</td>
<td>RECORD QUERIED</td>
<td>MINIMUM LOW</td>
</tr>
<tr>
<td>2/16</td>
<td></td>
<td>34.12</td>
<td>RECORD QUERIED</td>
<td>MINIMUM LOW</td>
</tr>
<tr>
<td>2/18</td>
<td></td>
<td>54.10</td>
<td>RECORD QUERIED</td>
<td>MAXIMUM HIGH</td>
</tr>
<tr>
<td>2/21</td>
<td></td>
<td>54.10</td>
<td>RECORD IN ERROR</td>
<td>MIN &gt; MAX</td>
</tr>
<tr>
<td>2/24</td>
<td></td>
<td>63.27</td>
<td>RECORD ADJUSTED</td>
<td>MAX TEMP NOT STATED</td>
</tr>
<tr>
<td>3/4</td>
<td></td>
<td>28.03</td>
<td>RECORD IN ERROR</td>
<td>MAXIMUM TOO HIGH</td>
</tr>
<tr>
<td>4/4</td>
<td></td>
<td>27.02</td>
<td>RECORD QUERIED</td>
<td>MAXIMUM HIGH</td>
</tr>
<tr>
<td>6/9</td>
<td></td>
<td>0.00</td>
<td>RECORD IN ERROR</td>
<td>MONTH INVALID</td>
</tr>
<tr>
<td>6/9</td>
<td></td>
<td>0.00</td>
<td>RECORD IN ERROR</td>
<td>MIN TEMP NOT STATED</td>
</tr>
</tbody>
</table>
## Attribute Table

<table>
<thead>
<tr>
<th>DCL NO.</th>
<th>IDENTIFIER</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>CARD</td>
<td>AUTOMATIC, STRING, CHARACTER</td>
</tr>
<tr>
<td>17</td>
<td>COMPLETE</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>2</td>
<td>DAY</td>
<td>AUTOMATIC, INITIAL, DECIMAL, FIXED(3,0)</td>
</tr>
<tr>
<td>25</td>
<td>DAYLIM</td>
<td>AUTOMATIC, DECIMAL, FIXED(3,0)</td>
</tr>
<tr>
<td>21</td>
<td>EDITOR</td>
<td>ENTRY, DECIMAL, FLOAT(SINGLE)</td>
</tr>
<tr>
<td>2</td>
<td>FALL</td>
<td>AUTOMATIC, INITIAL, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>2</td>
<td>FALLTOT</td>
<td>AUTOMATIC, INITIAL, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>2</td>
<td>FLAG</td>
<td>AUTOMATIC, BINARY, FIXED(15,0)</td>
</tr>
<tr>
<td>2</td>
<td>HRS</td>
<td>AUTOMATIC, INITIAL, DECIMAL, FIXED(3,0)</td>
</tr>
<tr>
<td>2</td>
<td>HRSSTOT</td>
<td>AUTOMATIC, INITIAL, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>2</td>
<td>IO_#</td>
<td>AUTOMATIC, BINARY, FIXED(15,0)</td>
</tr>
<tr>
<td>2</td>
<td>MAX</td>
<td>AUTOMATIC, INITIAL, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>25</td>
<td>MAXLIM</td>
<td>AUTOMATIC, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>2</td>
<td>MAXTOT</td>
<td>AUTOMATIC, INITIAL, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>2</td>
<td>MIN</td>
<td>AUTOMATIC, INITIAL, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>25</td>
<td>MINLIM</td>
<td>AUTOMATIC, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>2</td>
<td>MINTOT</td>
<td>AUTOMATIC, INITIAL, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>2</td>
<td>MONTH</td>
<td>AUTOMATIC, STRING, CHARACTER</td>
</tr>
<tr>
<td>53</td>
<td>Q1#001</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>61</td>
<td>Q1#002</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>73</td>
<td>Q1#003</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>219</td>
<td>Q1#004</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>365</td>
<td>Q1#005</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>381</td>
<td>Q1#006</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>389</td>
<td>Q1#007</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>73</td>
<td>Q2#001</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>Day</td>
<td>Month</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>MAX</td>
<td>MAX</td>
<td></td>
</tr>
<tr>
<td>MIN</td>
<td>MIN</td>
<td></td>
</tr>
<tr>
<td>FALL</td>
<td>FALL</td>
<td></td>
</tr>
<tr>
<td>MAX</td>
<td>MIN</td>
<td></td>
</tr>
<tr>
<td>MAX</td>
<td>MONTH</td>
<td></td>
</tr>
<tr>
<td>MAX</td>
<td>MONTH</td>
<td></td>
</tr>
<tr>
<td>MAX</td>
<td>MONTH</td>
<td></td>
</tr>
</tbody>
</table>

LOGICAL CROSS-REFERENCE TABLE FOR THIS EDIT IS
END WEATHER:

END OF THIS COMPILATION, NO ERRORS WERE NOTED IN THE EDITING STATEMENTS
PL/I F Compiler Options specified are as follows:
START, SIZE=108000, SM=(001,080)

The complete list of options used during this compilation is:

- EBCDIC
- CHAR40
- NOMACRO
- SOURCE2
- COMP
- SOURCE
- ATR
- NOXREF
- NOEXTREF
- NOLIST
- LOAD
- NODECK
- FLAGE
- SMRT
- SIZE=108000
- LINECNT=060
- OPT=60
- SORMGIN=(001,080)
DECLARE *** PROCEEDURE OPTIONS (MAIN): ***

1           WEATHER: PROCEDURE OPTIONS (MAIN):
           /* FIRST SAMPLE EDIT TO DEMONSTRATE FRED PRE-PROCESSOR
           EDITS TEMPERATURE AND RAINFALL CARDS */

2           DECLARE (MAX, MAXTOT, MIN, MINTOT, FALL, FALLTOT, HRTSTOT) FIXED DECIMAL (7, 2)
             CONTENTS (0),
             (DAY, HRS, TALLY) FIXED DECIMAL (3, 0) INITIAL (0),
             CARD CHARACTER (80),
             MONTH CHARACTER (3), FLAG FIXED BINARY;

3           ON ENDFILE (SYSIN) GO TO COMPLETE;
           PUT FILE (SYSPRINT) SKIP (60);

4           /* READ NEXT CARD AND SUBMIT TO EDIT */

5           READCARD: GET STRING (CARD) EDIT (CARD) (SYSCNT (80)):
           GET STRING (CARD) EDIT (CARD) (SYSCNT (80)):
           IF FLAG=3 THEN GO TO READDR; /* DO NOT USE ERRORS */

6           DECLARE CARD: FIXED DECIMAL (AX, 'TOT, INTOT, FALLT, TALLY, FIX: 7,2)
           INPVAL (0),
           (DAY, MINTOT, MAXTOT, FALL, MINTOT, MAX, MIN, FALL, HRS)
           FIXED DECIMAL (3, 0) INITIAL (0),
           TOATAL CHARACTER (3), FLAG FIXED BINARY;

7           ON ENDFILE (SYSIN) GO TO COMPLETE;
           PUT FILE (SYSPRINT) SKIP (60);

8           /* ALL CARDS READ, COMPUTE AND PRINT AVERAGES */

9           COMPLETE: MAXTOT=FALLTOT+MAX; MINTOT=MINTOT+MIN;
           FALLTOT=FALLTOT+FALL; HRTSTOT=HRTSTOT+HRS;

10          GO TO READDR;

11          /* INCLUDE EDIT AS INTERNAL PROCEDURE */

12          INCLUDE EDIT AS INTERNAL PROCEDURE:
           DECLARE Q #D (40) FIXED DECIMAL (7, 2)
           IGCT FIXED BINARY:
           DECLARE (Q #D MAX, Q #D MIN, Q #D FALL) FIXED DECIMAL (7, 2)
           DAM: FIXED DECIMAL (3, 0) ; Q #D MONTH CHARACTER (3) ; Q #D Q #D char
           FIXED DECIMAL (7, 2) , DAYLIM FIXED DECIMAL (3, 0) ; Q #D FATAL;

13          INQ Q #D QUERY, Q #D ADJUST, Q #D DIGCT=0; Q #D MAX = MAX ; Q #D MIN = MIN ;

14          IF HRS = 0 THEN:
15          IF HRS = 0 THEN:
16          IF HRS = 0 THEN:
17          IF HRS = 0 THEN:
18          IF HRS = 0 THEN:
19          IF HRS = 0 THEN:
20          IF HRS = 0 THEN:
21          IF HRS = 0 THEN:
22          IF HRS = 0 THEN:
23          IF HRS = 0 THEN:
24          IF HRS = 0 THEN:
25          IF HRS = 0 THEN:
26          IF HRS = 0 THEN:
27          IF HRS = 0 THEN:
28          IF HRS = 0 THEN:
29          IF HRS = 0 THEN:
30          IF HRS = 0 THEN:
31          IF HRS = 0 THEN:
32          IF HRS = 0 THEN:
33          IF HRS = 0 THEN:
34          IF HRS = 0 THEN:
35          IF HRS = 0 THEN:
36          IF HRS = 0 THEN:
37          IF HRS = 0 THEN:
38          IF HRS = 0 THEN:
39          IF HRS = 0 THEN:
40          IF HRS = 0 THEN:
41          IF HRS = 0 THEN:
42          IF HRS = 0 THEN:
43          IF HRS = 0 THEN:
44          IF HRS = 0 THEN:
45          IF HRS = 0 THEN:
46          IF HRS = 0 THEN:
47          IF HRS = 0 THEN:
48          IF HRS = 0 THEN:
49          IF HRS = 0 THEN:
50          IF HRS = 0 THEN:
51          IF HRS = 0 THEN:
52          IF HRS = 0 THEN:
53          IF HRS = 0 THEN:
54          IF HRS = 0 THEN:
55          IF HRS = 0 THEN:
56          IF HRS = 0 THEN:
57          IF HRS = 0 THEN:
58          IF HRS = 0 THEN:
59          IF HRS = 0 THEN:
60          IF HRS = 0 THEN:
61          IF HRS = 0 THEN:
62          IF HRS = 0 THEN:
63          IF HRS = 0 THEN:
64          IF HRS = 0 THEN:
65          IF HRS = 0 THEN:
66          IF HRS = 0 THEN:
67          IF HRS = 0 THEN:
68          IF HRS = 0 THEN:
69          IF HRS = 0 THEN:
70          IF HRS = 0 THEN:
71          IF HRS = 0 THEN:
72          IF HRS = 0 THEN:
73          IF HRS = 0 THEN:
74          IF HRS = 0 THEN:
75          IF HRS = 0 THEN:
76          IF HRS = 0 THEN:
77          IF HRS = 0 THEN:
78          IF HRS = 0 THEN:
79          IF HRS = 0 THEN:
80          IF HRS = 0 THEN:
81          IF HRS = 0 THEN:
82          IF HRS = 0 THEN:
83          IF HRS = 0 THEN:
84          IF HRS = 0 THEN:
85          IF HRS = 0 THEN:
86          IF HRS = 0 THEN:
87          IF HRS = 0 THEN:
88          IF HRS = 0 THEN:
89          IF HRS = 0 THEN:
90          IF HRS = 0 THEN:
WEATHER: PROCEDURE OPTIONS (MAIN):

I (DAY, MONTH, MAX, MIN, FALL, HRS) (8 (O, #FORM)); END; IF O. #DIGCT = 0 THEN DO: PUT ELS
E (SYSPRINT) EDIT (((O, #DIAG(IQ_#)) DO IQ_# = 1 TO O. #DIGCT)) (SKIP, A(40)); END; END
EDITOR; END WEATHER;
END WEATHER;

END OF THIS COMPILATION, ERRORS WERE NOTED AS SIGNALLED ABOVE.

DO NOT ATTEMPT TO RUN THIS PROGRAM UNTIL THESE ERRORS HAVE BEEN REMOVED.
DECLARE (MAX, MAXTOT, MIN, MINTOT, FALL, FALLTOT, HRSTOT) FIXED DECIMAL (7, 2)
INITIAL (0),
(DAY, HRS, TALLY) FIXED DECIMAL (3, 0) INITIAL (0),
CARD CHARACTER (80),
MONTH CHARACTER (3), FLAG FIXED BINARY;
ON ENDFILE (SYSIN) GO TO COMPLETE;
PUT FILE (SYSPRT) SKIP(60);
/* READ NEXT CARD AND SUBMIT TO EDIT */
READCARD: GET FILE (SYSIN) EDIT (CARD) (ABQ):;
GET STRING (CARD) EDIT (DAY, MONTH, MAX, MIN, FALL, HRS)
(F(2), A(3), 3 F(5, 2), F(2));
CALL EDITOR;
IF FLAG=3 THEN GO TO READCARD: /* DO NOT USE ERRORS */
TALLY=TALLY+1: MAXTOT=MAXTOT+MAX: MINTOT=MINTOT+MIN:
FALLTOT=FALLTOT+FALL: HRSTOT=HRSTOT+HRS:
GO TO READCARD:
/* ALL CARDS READ, COMPUTE AND PRINT AVERAGES */
COMPLETE: MAXTOT=MAXTOT/TALLY: MINTOT=MINTOT/TALLY:
HRSTOT=HRSTOT/TALLY:
PUT FILE (SYSPRT) EDIT ('RESULTS FOR ', 'TALLY', ' ACCEPTED',;
'RECORDS ARE ', 'AVERAGE MAXIMUM TEMP', 'MAXTOT',
'AVERAGE MINIMUM TEMP', 'MINTOT', 'TOTAL RAINFALL', 'FALLTOT',
'AVERAGE HOURS SUNSHINE', 'HRSTOT')
(PAGE=X(12), A(12), F(4), A(9), A(12), SKIP, A(20), X(5), F(7, 3),
SKIP, A(20), X(5), F(7, 3), SKIP, A(20), X(5), F(7, 3), SKIP,
A(22), X(3), F(7, 3));
/* INCLUDE EDIT AS INTERNAL PROCEDURE */
START EDIT EDITOR
EDIT (MAX, MIN, FALL) FIXED DECIMAL (7, 2), (DAY, HRS) FIXED DECIMAL
C(3, 0), MONTH CHARACTER (3)
FLAG FLAG;
LIST DAY, MONTH, MAX, MIN, FALL, HRS
FORMAT (SKIP, F(2), A(3), 3 F(5, 2), F(4))
MAXADJ 1
DECLARE (MAXLIM, MINLIM) FIXED DECIMAL (7, 2), DAYLIM FIXED DECIMAL
C(3, 0)
/* CHECK MONTH, AND GET NUMBER OF DAYS IN MONTH */
IF MONTH='JAN', 'MAR', 'MAY', 'JUL', 'AUG', 'OCT', 'DEC' THEN
DAYLIM=31
ELSE IF APR, 'JUN', 'SEP', 'NOV' THEN
DAYLIM=30
ELSE IF FEB THEN
DAYLIM=29
ELSE
ERROR 'MONTH INVALID'
GO TO TEST
/* TEST DAY */
IF DAY=0, DAYLIM THEN ERROR 'DAY INVALID'
ELSE
FRED PREF-PROCESSOR ***

/* CHECK THEN MAXIMUM TEMP IS PRESENT AND IN RANGE */
TST2 IF MIN>MAX THEN
IF MAX>0 THEN /* OMIT MAX NOT STATED FROM THIS TEST */
**
ELSE
ERROR 'MIN > MAX'
ELSE
IF MAX=0 THEN /* NOT STATED, INSERT A VALUE */
LOCK UP MONTH TO MAX,
C = 'JAN', 82.4, = 'FEB', 80.7, = 'MAR', 76.2, = 'APR', 66.7,
C = 'MAY', 59.3, = 'JUN', 52.6, = 'JUL', 51.8, = 'AUG', 95.1,
C = 'SEP', 81.4, = 'OCT', 67.0, = 'NOV', 72.9, = 'DEC', 92.2,
C OTHER 0, 'MAX TEMP NOT STATED'
ELSE /* GET UPPER LIMIT FOR TEST */
LOOK UP MONTH FOR MAXLIN,
C = 'JAN', 110.0, = 'FEB', 100.0, = 'MAR', 100.0, = 'APR', 90.0,
C = 'MAY', 75.0, = 'JUN', 65.0, = 'JUL', 55.0, = 'AUG', 72.0,
C = 'SEP', 92.0, = 'OCT', 90.0, = 'NOV', 102.0, = 'DEC', 105.0,
C OTHER 100.0
IF MAX-MAXLIN+5 THEN
ERROR 'MAXIMUM TOO HIGH'
M X LIN THEN
QUERY 'MAXIMUM HIGH'
ELSE
*/ CHECK THAT MINIMUM TEMP IS PRESENT AND IN RANGE */
IF MIN=0 THEN /* NOT STATED, INSERT A VALUE */
LOOK UP MONTH TO MIN,
C = 'JAN', 52.0, = 'FEB', 56.7, = 'MAR', 52.7, = 'APR', 45.5,
C = 'MAY', 30.1, = 'JUN', 35.0, = 'JUL', 35.0, = 'AUG', 35.0,
C = 'SEP', 38.9, = 'OCT', 44.2, = 'NOV', 48.7, = 'DEC', 53.3,
C OTHER 0, 'MIN TEMP NOT STATED'
ELSE /* GET LOWER LIMIT FOR TEST */
LOOK UP MONTH FOR MINLIN,
C = 'JAN', 38.0, = 'FEB', 35.0, = 'MAR', 35.0, = 'APR', 29.0,
C = 'MAY', 22.0, = 'JUN', 20.0, = 'JUL', 20.0, = 'AUG', 21.0,
C = 'SEP', 25.0, = 'OCT', 29.0, = 'NOV', 32.0, = 'DEC', 36.0,
C OTHER 25.0
IF MIN-MINLIN-5 THEN
ERROR 'MINIMUM TOO LOW'
MINLIN THEN
QUERY 'MINIMUM LOW'
ELSE
*/ CHECK RAINFALL */
IF FALL > 15 THEN
ERROR 'FALL TOO HIGH'
>5 THEN
QUERY 'FALL HIGH'
ELSE
*/ CHECK HOURS OF SUNSHINE */
IF HRS>20 THEN
ERROR 'HOURS TOO HIGH'
ELSE
**
END EDIT EDITOR
PROCEDURE:
    /* FRED PRE-PROCESSOR, DAVID MENDUS, 12/3/67
    TO SIGNAL AN UN-RECOGNISEABLE STATEMENT */

DECLARE (ENDFLAG FIXED BINARY(1),
        ERROR   FIXED BINARY(1)) STATIC EXTERNAL;

PUT FILE (SYSPRINT) EDIT ('*** STATEMENT TYPE CANNOT BE ',
                          'RECOGNISED') (SKIP,A(30),A(10));
ERROR=1;
LOOP:
    CALL ENTITY; IF ENDFLAG=1 THEN GO TO LOOP;
RETURN;

END XNST;
DECLARE (MAX,MAXTOT,MIN,MINTOT,FALL,FALLTOT,HRSSTOT) FIXED DECIMAL(7,2)
INITIAL (0);
(DAY,HRS,TALLY) FIXED DECIMAL (3,0) INITIAL (0),
CARD CHARACTER(80),
MONTH CHARACTER (3), FLAG FIXED BINARY;
ON ENDFILE (SYSIN) GO TO COMPLETE;
PUT FILE (SYSPRINT) SKIP(60);

READCARD: GET FILE (SYSIN) EDIT (CARD) (A(80));
GET STRING (CARD) EDIT (DAY,MONTH,MAX,MIN,FALL,HRS)
(F(2),A(3),3 F(5,2),F(2));
CALL EDITOR;
IF FLAG=3 THEN GO TO READCARD: /* DO NOT USE ERRORS */
TALLY=TALLY+1; MAXTOT=MAXTOT+MAX; MINTOT=MINTOT+MIN;
FALLTOT=FALLTOT+FALL; HRSSTOT=HRSSTOT+HRS;
GO TO READCARD;

COMPLETE: MAXTOT=MAXTOT/TALLY; MINTOT=MINTOT/TALLY;
HRSSTOT=HRSSTOT/TALLY;
PUT FILE (SYSPRINT) EDIT (RESULTS FOR 'TALLY', 'ACCEPTED',
'RECORDS ARE', 'AVERAGE MAXIMUM TEMP', MAXTOT,
'AVERAGE MINIMUM TEMP', MINTOT, 'TOTAL RAINFALL', FALLTOT,
'AVERAGE HOUR SUNSHINE', HRSSTOT)
(PAGE,X(2),A(12),F(4),A(9),A(12),SKIP,A(20),X(5),F(7,3),
SKIP,A(20),X(5),F(7,3),SKIP,A(14),X(11),F(7,3),SKIP,
A(22),X(3),F(7,3));

/* INCLUDE EDIT AS INTERNAL PROCEDURE */
START EDIT EDITOR
EDIT (MAX,MIN,FALL) FIXED DECIMAL (7,2), (DAY,HRS) FIXED DECIMAL
C(3,0), MONTH CHARACTER (3)
FLAG FLG.
LIST DAY,MONTH,MAX,MIN,FALL,HRS
FORMAT (SKIP,F(2),A(3),3 F(5,2),F(4))
MAXADJ 1
DECLARE (MAXLIM,MINLIM) FIXED DECIMAL (7,2), DAYLIM FIXED DECIMAL
C(3,0)
/* CHECK MONTH AND GET NUMBER OF DAYS IN MONTH */
IF MONTH='JAN',='MAR',='MAY',='JUL',='AUG',='OCT',='DEC' THEN
DAYLIM=31
= 'APR',='JUN',='SEP',='NOV' THEN
DAYLIM=30
= 'FEB' THEN
DAYLIM=29
IF DAY=0, DAYLIM THEN /* TEST DAY */
*** ELSE STATEMENT TO IF-GROUP CLOSED BY THIS STATEMENT
ERROR 'DAY INVALID'
*** STARTING COLUMN IS ILLEGAL
ELSE
*** STARTING COLUMN IS ILLEGAL
FRED PRE-PROCESSOR **

/* CHECK THEN MAXIMUM TEMP IS PRESENT AND IN RANGE */
TST2 IF MIN>MAX THEN
IF MAX=0 THEN /* OMIT MAX NOT STATED FROM THIS TEST */
**
ELSE
**
ERROR 'MIN > MAX'
ELSE
**
IF MAX=0 THEN /* NOT STATED, INSERT A VALUE */
LOOK UP MONTH TO MAX,
C = 'JAN', 82.4, = 'FEB', 80.7, = 'MAR', 76.2, = 'APR', 66.7,
C = 'MAY', 59.3, = 'JUN', 52.8, = 'JUL', 51.8, = 'AUG', 55.1,
C = 'SEP', 61.4, = 'OCT', 67.0, = 'NOV', 72.9, = 'DEC', 79.2,
C = OTHER 0, 'MAX TEMP NOT STATED'
ELSE /* SET UPPER LIMIT FOR TEST */
LOOK UP MONTH FOR MAXILIM,
C = 'JAN', 110.0, = 'FEB', 100.0, = 'MAR', 100.0, = 'APR', 90.0,
C = 'MAY', 75.0, = 'JUN', 65.0, = 'JUL', 65.0, = 'AUG', 72.0,
C = 'SEP', 82.0, = 'OCT', 90.0, = 'NOV', 102.0, = 'DEC', 103.0,
C = OTHER 108.0
IF MAX>MAXILIM+5 THEN
ERROR 'MAXIMUM TOO HIGH'
MAXILIM THEN
QUERY 'MAXIMUM HIGH'
ELSE
**
/* CHECK THAT MINIMUM TEMP IS PRESENT AND IN RANGE */
IF MIN=0 THEN /* NOT STATED, INSERT A VALUE */
LOOK UP MONTH TO MIN,
C = 'JAN', 36.0, = 'FEB', 56.1, = 'MAR', 52.7, = 'APR', 45.5,
C = 'MAY', 39.1, = 'JUN', 38.7, = 'JUL', 33.6, = 'AUG', 35.4,
C = 'SEP', 38.9, = 'OCT', 44.2, = 'NOV', 48.7, = 'DEC', 53.3,
C = OTHER 0, 'MIN TEMP NOT STATED'
ELSE /* SET LOWER LIMIT FOR TEST */
LOOK UP MONTH FOR MINILIM,
C = 'JAN', 38.0, = 'FEB', 35.0, = 'MAR', 35.0, = 'APR', 29.0,
C = 'MAY', 22.0, = 'JUN', 18.0, = 'JUL', 20.0, = 'AUG', 21.0,
C = 'SEP', 25.0, = 'OCT', 29.0, = 'NOV', 32.0, = 'DEC', 36.0,
C = OTHER MINILIM
ERROR 'MINIMUM TOO LOW'
MINILIM THEN
QUERY 'MINIMUM LOW'
ELSE
**
/* CHECK RAINFALL */
IF FALL > 15 THEN
ERROR 'FALL TOO HIGH'
THEN
QUERY 'FALL HIGH'
ELSE
**
END EDIT EDITOR .
<table>
<thead>
<tr>
<th></th>
<th>MAX</th>
<th></th>
<th>MIN</th>
<th></th>
<th>MONTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALL</td>
<td>****</td>
<td>FALL</td>
<td>MIN</td>
<td></td>
<td>DAY</td>
</tr>
<tr>
<td></td>
<td>****</td>
<td>MAX</td>
<td></td>
<td></td>
<td>MONTH</td>
</tr>
<tr>
<td></td>
<td>****</td>
<td>MAX</td>
<td>MIN</td>
<td></td>
<td>MONTH</td>
</tr>
</tbody>
</table>
STARTER: PROCEDURE;

DECLARE (ENDFLAG FIXED BINARY(1),
ENDPROG FIXED BINARY(1),
EDITNAM CHARACTER(7) STATIC EXTERNAL,
NAMHOLD(7) PACKED CHARACTER(1) DEFINED EDITNAM,
CARD(90) PACKED CHARACTER(1),
KEY CHARACTER(16) DEFINED CARD,
ALPHABET(29) PACKED CHARACTER(1) DEFINED ALPHA,
ALPHA CHARACTER(29) STATIC INITIAL
('ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890');

READ CARDS UNTIL START EDIT STATEMENT /*
ON ENDFILE (SYSIN) BEGIN;
GO TO ENDIT; END;
LOOP:
GET FILE (SYSIN) EDIT (CARD) (80 A(1));
IF KEY=' START EDIT THEN GO TO FOUND;
PUT FILE (OBJECT) EDIT (CARD(1)) DO I=2 TO 72). (71 A(1));
PUT FILE (SYSPRINT) EDIT (CARD) (SKIP, 80 A(1));
GO TO LOOP;
FOUND:
EXTRACT EDIT NAME /*
PUT FILE (SYSPRINT) EDIT (CARD) (SKIP, 80 A(1));
IF KEY=' END EDIT THEN GO TO END;
PUT FILE (SYSPRINT) EDIT (' *** NO NAME HAS BEEN GIVEN TO ',
'THIS EDIT') (SKIP, A(31), A(9)); ERROR=1; RETURN;
NAME:
NAME MUST HAVE ALPHA FIRST CHARACTER, NOT EXCEED 7 CHARS /*
PUT FILE (SYSPRINT) EDIT (' *** EDIT NAME DOES NOT HAVE ',
'ALPHABETIC FIRST CHARACTER!') (SKIP, A(29), A(26));
ERROR=1; RETURN;
OK:
J=1 TO 7;
NAMHOLD(J)=CARD(I+J-1);
IF (CARD(I+J)=' ')(CARD(I+J)=' ') THEN GO TO ENDNAM;
END;
PUT FILE (SYSPRINT) EDIT (' *** EDIT NAME IS TOO LONG')
(SKIP, A(26)); ERROR=1; RETURN;
ENDNAM:
J=1 TO 7; NAMHOLD(J)=' '; END; /* PAD WITH BLANKS */
I=1; // RESET I FOR PUT STATEMENT */
/* WRITE PROCEDURE STATEMENT INCLUDING PARAMETERS */
PUT FILE (OBJECT) EDIT (EDITNAM,' PROCEDURE !,(CARD(J) DO
J=1 TO 72),+1) (A(7), A(11), 70 A(1));
/* WRITE DECLARE FOR PRIV,TE FLAGS */
PUT FILE (OBJECT) EDIT

('DECLARE Q adjust Q 0 Q diag(8) PACKED CHARACTER(32)',
'(Q # FATAL Q # QUERY) FIXED BINARY(1)',
'(Q # ADJUST Q # DIAG) FIXED BINARY(1)') (A(41), A(36), A(34));
ENDPROG=0; RETURN;
ENDIFIT: ENDPUSG=1; RETURN;
END STARTER;
PROCEDURE: /* FRED PRE-PROCESSOR, DAVID WNEUDUS, 24/3/67, TO PROCESS A PRINTALL STATEMENT */

DECLARE (ENDFLG  FIXED BINARY(1),
PRALFLG  FIXED BINARY(1) ) STATIC EXTERNAL;

IF PRALFLG THEN DO: /* ONLY ONE PRINTALL STATEMENT */
PUT FILE (SYSPRINT) EDIT ('*** THIS IS NOT THE FIRST ',
*PRINTALL STATEMENT*) (SKIP,A(27),A(18)); ERROR=1:
GO TO BADEND;
END:
PRALFLG=1:
CALL ENTITY; IF ENDFLAG THEN RETURN; /* NO MORE LOCAL */
PUT FILE (SYSPRINT) EDIT ('*** PRINTALL STATEMENT FORMAT ',
*INVALID*) (SKIP,A(31),A(7)); ERROR=1:
BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
END PRST;
STST:  PROCEDURE:
/* FRED PRE-PROCESSOR, DAVID MENDUS, 24/3/67
   TO EDIT AND PROCESS A START SUBEDIT STATEMENT */

DECLARE (ENDFLAG, FIXED BINARY(1),
SYNTAX(32) PACKED CHARACTER(1),
SYNCT Ny, FIXED BINARY1),
ERROR FIXED BINARY(1),
SUBFLG FIXED BINARY(1),
SUBNAME(32) PACKED CHARACTER(1),
SUBCNT FIXED BINARY,)
COLONE FIXED BINARY,)
STATIC EXTERNAL,
ALPHA CHARACTER(29) STATIC INITIAL
( 'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
ALPHA CHARACTER(32) DEFINED ALPA,
WORD CHARACTER(32) DEFINED SYNTAX:

IF COLONE=7 THEN DO: /* MUST BE IN FIRST COL OF MATRIX */
   PUT FILE (SYSPRINT) EDIT (' *** START STATEMENT MUST BEGIN',
   IN COLUMN 7') (SKIP,A(31),A(12)) ; ERROR=1; GO TO BADEND;
END:

CALL ENTITY; /* NEXT STATEMENT MUST BE SUBEDIT */

IF ENDFLAG THEN GO TO TOOSOON;
IF WORD='SUBEDIT' THEN GO TO BADFORM:
IF SUBFLG THEN DO: /* NESTED SUBEDITS ARE ILLEGAL */
   PUT FILE (SYSPRINT) EDIT (' *** THIS STATEMENT OCCURS IN',
   'A SUBEDIT, DELETED') (SKIP,A(30),A(18)) ; ERROR=1;
   GO TO BADEND; END:

CALL ENTITY; /* NEXT UNIT IS SUBEDIT NAME */
IF ENDFLAG THEN GO TO TOOSOON:
IF SYNCT Ny THEN GO TO RADNAME; /* CHECK VALID NAME FORMAT */
DO i=1 TO 29; IF SYNTAX(i)=ALPHABET(1) THEN GO TO OK; END:
BADNAME: PUT FILE (SYSPRINT) EDIT (' *** FORMAT OF SUBEDIT NAME IS',
'INVALID') (SKIP,A(31),A(7)) ; ERROR=1; GO TO BADEND:

OK:
SUBFLG[i]=1; SUBNAME=SYNTAX; SUBCNT=SYNCNT; /* STORE NAME */
PUT FILE (OBJECT) EDIT (SYNTAX(i) DO i=1 TO SYNGCN)!
(X(i),A(11)); PUT FILE (OBJECT) EDIT (': PROCEDURE;')
(A(12));

BADFORM: /* SHOULD BE NO MORE */
IF ENDFLAG THEN RETURN;
BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN:
TOOSOON: PUT FILE (SYSPRINT) EDIT (': *** START SUBEDIT STATEMENT',
'IS INCOMPLETE') (SKIP,A(29),A(13));

END STST:
DO I=1 TO 6; IF WORD=SPECIAL(I) THEN GO TO OK3; END;
GO TO BADFORM; /* NOT LOGICAL OPERATOR */

OK3:
PUT FILE (OBJECT) EDIT (' IF ',(HOLD1(I) DO I=1 TO CNT1),
(SYNTAX(I) DO I=1 TO SYNCNT)) (A(4),34 A(1));
LOOP1:
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON; /* END OF FIRST EXP */
IF SYNTAX(I)='.' THEN GO TO SECOND; /* END OF FIRST EXP */
PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
I F SYNTAX(I)='.' THEN CALL BRACKET; GO TO LOOP1;
SECOND:
/* PROCESS SECOND EXPRESSION OF PAIR */
IF TYPEFLG=-1 THEN GO TO GETLAB; /* IF NOT LABEL FORM ASS. */
PUT FILE (OBJECT) EDIT (' THEN DO:/', {A(9)});
PUT FILE (OBJECT) EDIT ((HOLD2(I) DO I=1 TO CNT2))
(X(1),32 A(1));
LOOP2:
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF SYNTAX(I)='.' THEN CALL BRACKET; /* NOW COPY REST */
PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
(X(1),32 A(1));
IF SYNTAX(I)='.' THEN CALL BRACKET;
FINISH:
PUT FILE (OBJECT) EDIT ('; GO TO ',UPLAB,'; END;')
(A(8),A(7),A(6)); GO TO MAINLOOP;
/* JUMP, READ LABEL AND WRITE JUMP */
GETLAB:
PUT FILE (OBJECT) EDIT (' THEN DO:/', {A(12)});
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
BADLAB:
DO I=1 TO 29; IF SYNTAX(I)=ALPHABET(I) THEN GO TO OKLAB; END;
/* LABEL GIVEN IN LIST HAS */
'INVALID FORMAT' (SKIP,A(29),A(14)); ERROR=1; GO TO BADEND;
OKLAB:
PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT),';');
(A(8),)
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF SYNTAX(I)=',' THEN GO TO MAINLOOP; ELSE GO TO BADLAB;
OTHER:
IF TYPEFLG=-1 THEN GO TO LASTLAB;
PUT FILE (OBJECT) EDIT ((HOLD2(I) DO I=1 TO CNT2))
(X(1),32 A(1));
PUT FILE (OBJECT) EDIT (' THEN DO:/', {A(12)});
LASTLOOP:
CALL ENTITY; IF ENDFLAG THEN GO TO ONEGOOD;
IF SYNTAX(I)='.' THEN GO TO NILGOOD;
PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
(X(1),32 A(1));
IF SYNTAX(I)='.' THEN CALL BRACKET;
PRINT:
GO TO LASTLOOP;
ONEGOOD:
IF TYPEFLG=-1 THEN GO TO TOOSOON; GO TO WRITLAST;
NILGOOD:
IF TYPEFLG=0 THEN GO TO BADFORM;
WRITLAST:
PUT FILE (OBJECT) EDIT (';UPLAB;') (A(1)),A(7),A(1));
UPCNT=UPCNT+1; /* UPDATE LABEL FOR THIS STATEMENT */
PUT STRING (UPLAB) EDIT ('_#ADJUST#_ADJUST+1;') (A(23));
RETURN;
/* PROCESS END OF JUMP STATEMENT */
LASTLAB:
PUT FILE (OBJECT) EDIT ('; GO TO ') (A(6));
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSON;
DO I=1 TO 29; IF SYNTAX(I)=ALPHABET(I) THEN GO TO OKZ; END;
GO TO BADLAB;
OKZ:
PUT FILE (OBJECT) EDIT {SYNTAX(I) DO I=1 TO SYMCNT},';';
(X(1),33 A(1));
CALL ENTITY; IF ENDFLAG THEN RETURN;

/* GENERAL ERROR ROUTINE */
BADFORM: PUT FILE (SYSPRINT) EDIT (' *** FORMAT OF LOOK-UP INVALID')
(SKIP, A(30)); ERROR=1;
BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
TOOSON: PUT FILE (SYSPRINT) EDIT (' *** LOOK-UP STATEMENT INCOMPLETE')
(SKIP, A(33)); ERROR=1; RETURN;
END LOST:
PROCEDURE: PROCED PRE-PROCESSOR. DAVID MENDUS, 26/3/67
TO PROCESS A MAXADJ STATEMENT */

DECLARE (ENDFLAG FIXED BINARY(1),
SYNTAX(32) PACKED CHARACTER(1),
SYNENT FIXED BINARY,
ERROR FIXED BINARY(1),
MAXVAL FIXED BINARY ) STATIC EXTERNAL,
NUM CHARACTER(9) INITIAL ('123456789') STATIC,
NUMBERS(9) PACKED CHARACTER(1) DEFINED NUM;

/* ONLY UNIT SHOULD BE FIXED POINT CONSTANT */
CALL ENTITY;
IF ENDFLAG THEN DO:
   PUT FILE (SYSPRINT) EDIT (' *** MAXADJ STATEMENT IS ',
   'INCOMPLETE') (SKIP,A(25),A(10)); ERROR=1; RETURN; END;
IF SYNENT=1 THEN GO TO BADEND;
DO i=1 TO 9; IF SYNTAX(i)=NUMBERS(i) THEN GO TO OK; END;
GO TO BADEND;

OK: GET STRING (SYNTAX(1)) EDIT (MAXVAL) (F(1,0));
CALL ENTITY; IF ENDFLAG THEN RETURN;
BADEND: PUT FILE (SYSPRINT) EDIT (' *** NUMBER IS NOT INTEGER 1-9')
   (SKIP,A(30)); ERROR=1;
BAD: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BAD; RETURN;
END MAST;
NEXTONE: PROCEDURE;  
/* FRED PRE-PROCESSOR, DAVID MENDUS, 19/2/67 */  
PART OF THE RECOGNISER TO RETURN THE NEXT SIGNIFICANT  
CHARACTER, PROCESSES CONTINUATION CARDS AND READS OVER  
COMMENTS.  

DECLARE  
(COLCT FIXED BINARY;  
COLONE FIXED BINARY;  
ENDFLG FIXED BINARY(1);  
INWORD FIXED BINARY(1);  
REPEAT FIXED BINARY(1);  
ERROR FIXED BINARY(1);  
INTEND FIXED BINARY(1) INITIAL(0);  
CHARA CHARACTER(1) I Static EXTERNAL,  

(COMMENT FIXED BINARY(1) INITIAL(0);  
APOSFLG FIXED BINARY(1) INITIAL(0);  
CARD PACKED CHARACTER(1);  
ALPHABET CHARACTER(41) INITIAL  
("ABCDFGHJKLMNPQRSTUVWXYZ@#",..0123456789")  
) Static;  

SCANLIST(41) PACKED CHARACTER(1) DEFINED ALPHABET:  

/*-----*/  
IF INTEND THEN DO:  
INTEND=0; ENDFLAG=1; RETURN; END;  
IF REPEAT THEN DO:  
/* IF REPEAT FLAG IS SET, DO - */  
REPEAT=0; END;  
/* NOT INCREMENT COLUMN COUNT */  
ELSE DO:  
IF COLCNT=0 THEN DO:  
/* IF NEW CARD, THEN PRINT IT */  
PRINT:  
PUT FILE (SYSPRINT) EDIT ((CARD(1) DO I=1 TO 80))  
(SKIP,X(11),B0A(11));  
COLONE=0; END;  
COLCNT=COLCNT+1; END;  
/* INCREMENT COLUMN COUNT */  
IF COLCNT>73 THEN GO TO NEXTCHAR;  

NEWCARD:  
GET FILE (SYSSIN) EDIT ((CARD(1) DO I=1 TO 80)) (B0A(11));  
IF CARD(I)='' THEN DO:  
/* NOT CONTINUATION CARD */  
IF COLONE=0 THEN DO:  
/* LAST CARD WAS NOT SIGNIFICANT */  
COLN0=0; GO TO PRINTIM; /* DO NOT RETURN */  
END;  
IF INWORD=0 THEN DO:  
/* DO NOT SIGNAL END OF CARD YET */  
INWORD=INTEND=CHARA=''; RETURN; END;  
ENDFLAG=1; /* SIGNAL END OF STATEMENT */  
INWORD=0; COMMENT=0; COLCNT=0;  
RETURN; END;  

PUT FILE (SYSPRINT) EDIT ((CARD(1) DO I=1 TO 80))  
(SKIP,X(11),B0A(11));  
/* PRINT CONTINUATION CARD */  
DO COLCNT=1 TO 7 TO 72;  
/* FIND FIRST NON-BLANK COLUMN */  
IF CARD(COLCNT)='' THEN GO TO FOUND;  
END;  
IF COLCNT=COLONE THEN GO TO NEXTCHAR;  
IF COLONE=0 THEN DO:  
/* IF WE ARE IN CONTINUED COMMENT */  
IF COLCNT=73 THEN GO TO NEWCARD; /* THEN COLONE WON'T BE SET */  
ELSE GO TO NEXTCHAR; /* DO NOT FLAG ERROR */  
END;  
/* LOOP IGNORING ALL SUCCEEDING CONTINUATION CARDS */  
REJECT:  
PUT FILE (SYSPRINT) EDIT (' ** ILLEGAL CONTINUATION CARD',":"
*CONTENTS IGNORED*) (SKIP,A(32),A(16));
ERROR=1;
GET FILE (SYSIN) EDIT (CARD(1) DO I=1 TO 80) (80A(1));
IF CARD(6)=*** THEN GO TO ENDTCA: /*SIGNAI END OF STATEMENT */
PUT FILE (SYSPRINT) EDIT (CARD(1) DO I=1 TO 80)
{SK1P,X(1),80A(1)};
GO TO REJECT;

/* STORE AND PROCESS NEXT CHARACTER */
NEXTCHAR:CHARA=CARD(COLCNT);
/*STORE CHARACTER FOR RETURN */
IF AP0SFLG THEN DO:
IF CHARA= Then APOSFLG=0;
INWORD=1; RETURN; END;
IF CHARA= ' THEN DO:
/*PROCESS A BLANK CHARACTER */
IF INWORD=0 THEN GO TO INCRCT; /*IF EXCESS BLANK, IGNORE */
ELSE DO:
INWORD=0; RETURN; END; /*SIGNAL BLANK AND RETURN */
END:

/* CHECK FOR START OR FINISH OF A COMMENT */
IF COMMENT THEN DO: /* IN A COMMENT */
IF CHARA= ' THEN GO TO INCRCT; /* NOT END; IGNORE */
IF COLCNT=77 THEN GO TO INCRCT; /*COMMENT CANNOT END ->
COLCNT=COLCNT+1; /*OVER CARD BOUNDARY */
IF CHARA(COLCNT)-=' THEN GO TO NEXTCHAR;
COMMENT=0; GO TO INCRCT; /*END OF COMMENT FOUND*/
END:
IF CHARA= ' THEN DO: /*LOOK FOR THE START OF COMMENT */
IF COLCNT=77 THEN GO TO NOCOMM;
IF CHARA= ' THEN GO TO NOCOMM;
COMMENT=1; /*START OF COMMENT */
COLCNT=COLCNT+1;
IF INWORD=0 THEN GO TO INCRCT; /*IF PRECEDED BY BLANKS DO->
INWORD=0; CHARA= ' ; /*NOT SIGNAL, ELSE SIGNAL */
RETURN: /*BLANK (END OF WORD) */
END:

/*DETERMINE TYPE OF CHARACTER AND SIGNAL ACCORDINGLY */
NOCOMM: DO I=1 TO 41;
IF CHARA=ALNHIST(1) THEN DO;
INWORD=1; RETURN; END;
/*SIGNAL ALPHAMERIC */
END:
IF INWORD=1 THEN DO: /*SPECIAL CHARACTER */
IF CHARA=*** THEN DO:
AP0SFLG=1; INWORD=1; RETURN; END;
INWORD=1; RETURN; END; /*IF OUTSIDE WORD, SIGNAL */
INWORD=0; CHARA= '; /*IF INSIDE WORD SIGNAL DUMMY */
REPEAT=1; RETURN; /*BLANK AND SET REPEAT FLAG */
END NEXTCHAR;
PEST:  PROCEDURE;  /* FRED PRE-PROCESSOR, DAVID MENDUS, 24/3/67  
TO PROCESS A PERFORM STATEMENT */

DECLARE (ENDFLAG FIXED BINARY(1),  
SYNTAX(32) PACKED CHARACTER(1),  
SYNCNT FIXED BINARY,  
ERROR FIXED BINARY(1) STATIC EXTERNAL,  
ALPHA CHARACTER(29) STATIC INITIAL  
(*ABCDEFHJKLMNOPQRSTUVWXYZ#@$1,  
ALPHABET(29) PACKED CHARACTER(1) DEFINED ALPHA;  
CALL ENTITY; /* NEXT UNIT SHOULD BE SUBEDIT NAME */
IF ENDFLAG THEN DO;  
PUT FILE (SYSPRINT) EDIT ('*** NO SUBEDIT IS SPECIFIED')  
(SKIP,A(29)); ERROR=1; RETURN; END;
IF SYNCYN<8 THEN DO: /* CHECK FORMAT OF SUBEDIT NAME */
DO I=1 TO 29; IF SYNTAX(I)=ALPHABET(I) THEN GO TO OK; END;
END: /* NAME FORMAT BAD */
PUT FILE (SYSPRINT) EDIT ('*** SUBEDIT NAME FORMAT INVALID')  
(SKIP,A(32)); ERROR=1; GO TO BADEND;

OK:  PUT FILE (OBJECT) EDIT ('CALL',SYNTAX(I) DO I=1 TO SYNCYN),  
(':')(A6),8 A(I));  
CALL ENTITY; IF ENDFLAG THEN RETURN; /* SHOULD END HERE */
PUT FILE (SYSPRINT) EDIT ('*** PERFORM STATEMENT FORMAT ',  
'INVALID') (SKIP,A(30),A(7)); ERROR=1;
BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
END PEST;
/* FRED PRE-PROCESSOR, DAVID MENDUS, 12/3/67 */
TO PROCESS A PRINTNONE STATEMENT

DECLARE (ENDFLAG FIXED BINARY(1),
    ERROR FIXED BINARY(1),
    LISTFLG FIXED BINARY(1)) STATIC EXTERNAL;

/* THERE MAY BE ONLY ONE LIST OR PRINTNONE STATEMENT */
IF LISTFLG=0 THEN DO:
    PUT FILE (SYSPRINT) EDIT ('*** THIS IS NOT THE FIRST ',
        'LIST OR PRINTNONE STATEMENT') (SKIP, A(27), A(27));
    ERROR=1: END;
ELSE LISTFLG=1;
    CALL ENTITY; IF ENDFLAG=1 THEN RETURN;
    PUT FILE (SYSPRINT) EDIT ('*** PRINTNONE FORMAT ILLEGAL')
        (SKIP, A(29)): ERROR=1;
LOOP: CALL ENTITY; IF ENDFLAG=0 THEN GO TO LOOP;
RETURN;
END_PNST:
LAST: PROCEDURE:
/* FRED PRE-PROCESSOR, DAVID MENDUS; 16/3/67 */
TO PROCESS A LABEL TO A STATEMENT */
DECLARE (SYNTAX(32) PACKED CHARACTER(1),
SYNCNT FIXED BINARY ) STATIC EXTERNAL;
PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT),"
(X(I),6 A(I));
END LAST;
LIST: PROCEDURE:
/* FRED PRE-PROCESSOR, DAVID MENDUS, 27/3/67
TO PROCESS A LIST STATEMENT */

DECLARE (ENDFLAG FIXED BINARY(1),
SYNTAX(32) PACKED CHARACTER(1),
SYNCT FIXED BINARY,
ERROR FIXED BINARY(1),
LISTFLG FIXED BINARY(1),
LISTCNT FIXED BINARY ) STATIC EXTERNAL,

LISTSTR(LISTCNT) PACKED CHARACTER(1) CONTROLLED EXTERNAL,
LISTTEMP(2000) PACKED CHARACTER(1):
IF LISTFLG THEN DO: /* ONLY ONE LIST OR PRINTNONE ALLOWED */
PUT FILE (SYSPRINT) EDIT ("*** THIS IS NOT THE FIRST ",
"LIST OR PRINTNONE STATEMENT") (SKIP,A(27),A(27));
ERROR=1: GO TO BADEND; END;
LISTFLG=1: LISTCNT=0: /* INITIALISE COUNT */

LOOP: CALL ENTITY: IF ENDFLAG THEN DO: /* MOVE LIST TO EXT.STORE */
IF LISTCNT+SYNCT>2000 THEN DO: /* NO ROOM TO STORE UNIT */
PUT FILE (SYSPRINT) EDIT (" *** LENGTH OF LIST STATEMENT ",
"EXCEEDS COMPILER LIMIT") (SKIP,A(30),A(23)); ERROR=1;
BADEND: CALL ENTITY: IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
END;
DO I=1 TO SYNCT;
LISTTEMP(LISTCNT+I)=SYNTAX(I);
END:
LISTCNT=LISTCNT+SYNCT; GO TO LOOP:

END LIST:
PROCEDURE:
// FRED PRE-PROCESSOR, DAVID MENDUS, 2/4/67
// PROCESS LOOK UP STATEMENTS OF THREE KINDS. LOOK UP JUMP,
// LOOK UP TO LOOK UP BY CALLS ACON TO COMPLETE A ... TO
// STATEMENT WITH DIAGNOSTIC ACTION. */
DECLARE (ENDFLAG) FIXED BINARY(1),
SYNTAX(32) PACKED CHARACTER(1),
SYNCNT FIXED BINARY,
ERROR FIXED BINARY(1),
NITEM FIXED BINARY ) STATIC EXTERNAL,
RECORD(NITEM) PACKED CHARACTER(12) CONTROLLED EXTERNAL;
(UPLAB CHARACTER(7) INITIAL('O #LO01'),
UNCRN FIXED BINARY-INITIAL(1001),
TYPEFLG FIXED BINARY(1),
HOLD1(32) PACKED CHARACTER(1),
CNT1 FIXED BINARY,
CNT2 FIXED BINARY,
SP CHARACTER(12) INITIAL('>' < = >= <= =) )
STATIC;
ALPHA CHARACTER(29) STATIC INITIAL
('ABCDEFGHIJKLMNOPQRSTUVWXYZ',
SPECIAL(6) PACKED CHARACTER(2) DEFINED SP,
ALPHAT(29) PACKED CHARACTER(1) DEFINED ALPHA,
WORD CHARACTER(32) DEFINED SYNTAX;
/* STORE FIRST NAME AND DECIDE TYPE OF LOOKUP STATEMENT */
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSCON;
IF (SYNTAX(1)='F') & (SYNTAX(2)='O') & (SYNTAX(3)='R') THEN TYPEFLG=1; GO TO BADFORM; /* TYPE IS JUMP, NO SECOND NAME */
IF (SYNTAX(1)='T') THEN TYPEFLG=1; GO TO BADFORM; /* TYPE IS JUMP, NO SECOND NAME */
IF (SYNTAX(1)='F') & (SYNTAX(2)='O') THEN TYPEFLG=1; GO TO BADFORM; /* TYPE IS JUMP, NO SECOND NAME */
IF (SYNTAX(1)='T') THEN TYPEFLG=1; GO TO BADFORM; /* TYPE IS JUMP, NO SECOND NAME */
READONE: CALL ENTITY; IF ENDFLAG THEN GO TO TOOSCON;
IF (TYPEFLG=0) THEN GO TO OK1; END;
PUT FILE (SYSPRNT) EDIT (" *** FIRST VARIABLE NAME FORM AT ",
'ILLEGAL') (SKIP,A(32),A(7)): ERROR=1; GO TO BADEND;
BEGIN: /* STORE FIRST NAME */
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSCON;
IF (SYNTAX(1)='F') & (SYNTAX(2)='O') THEN TYPEFLG=1; GO TO BADFORM; /* TYPE IS JUMP, NO SECOND NAME */
IF (SYNTAX(1)='T') IF (SYNTAX(2)='O') THEN TYPEFLG=1; GO TO BADFORM; /* TYPE IS JUMP, NO SECOND NAME */
READONE: CALL ENTITY; IF ENDFLAG THEN GO TO TOOSCON;
IF (TYPEFLG=1) THEN GO TO OK1; END;
PUT FILE (SYSPRNT) EDIT (" *** SECOND VARIABLE IS NOT ",
PART OF RECORD) (SKIP,A(29),A(41)): ERROR=1; GO TO BADEND;
OK1: Hold2-SYNTAX: CNT2-SYNTAX: /* STORE SECOND NAME (NOT JUMP) */
LOOKCOM: CALL ENTITY; IF ENDFLAG THEN GO TO TOOSCON;
IF WORD=*, THEN GO TO BADFORM;
/* START DECODING NEXT PAIR OF LOOKUP */
MAINLOOP: CALL ENTITY; IF ENDFLAG THEN GO TO TOOSCON;
IF WORD=OTHER THEN GO TO OTHER;
KEYWORD: PROCEDURE:
/# FRED PRE-PROCESSOR, DAVID MENDUS, 19/2/67
PART OF THE RECOGNISER TO BUILD AND IDENTIFY THE KEYWORD
FOR EACH FRED STATEMENT. */

DECLARE (COLCNT FIXED BINARY,
COLONE FIXED BINARY,
ENDFLAG FIXED BINARY(1),
INWORD FIXED BINARY(1),
REPEAT FIXED BINARY(1),
SYNTAX(32) PACKED CHARACTER(1),
SYNCNT FIXED BINARY,
WORD CHARACTER(32) DEFINED SYNTAX,
RECORD(NITEM) PACKED CHARACTER(12) CONTROLLED EXTERNAL,
FLAG(ITEM) PACKED BINARY(1) CONTROLLED EXTERNAL,
ITEM FIXED BINARY STATIC EXTERNAL,
ALF CHARACTER(29) STATIC INITIAL
"ABCDEFGHIJKLMNOPQRSTUVWXYZ",
ALPHABET(29) PACKED CHARACTER(1) DEFINED ALF;

CL: CLEAR FLAGS AND READ FIRST CHARACTER */
RESTART: WORD=0;
ENDFLAG=0;
CALL NEXTONE:
IF INWORD=0 THEN GO TO RESTART;
IF COLCNT=5 THEN GO TO STMNT;

/* FIRST CHARACTER IS PART OF LABEL */
DO I=1 TO 29; /* TEST FOR ALPHABETIC FIRST CHARACTER */
IF CHARA ALPHABET(1) THEN GO TO ALPHST;
END:
/* ERROR MESSAGE IF NOT ALPHA FIRST CHARA */
FILE SYSPRINT EDIT (' *** LABEL FORMAT INVALID',
'LABEL HAS BEEN DELETED', (SKIP, M(27), M(22)));
ERROR=1;
/* FIX COUNTS TO IGNORE LABEL AND RESTART */
ALPHST: SYNTAX=SYNCNT+1; SYNTAX(SYNTAX)=CHARA;
CALL NEXTONE:
IF INWORD THEN GO TO ALPHST; /* LOOP UNTIL LABEL ASSEMBLED */
FLAG=29: RETURN; /* SET LABEL FLAG AND RETURN */

STMNT: COLONE=COLCNT; /* SET STARTING COLUMN FLAG */
SYNCNT=1; SYNTAX(1)=CHARA;
IF INWORD=0 THEN GO TO NOSPEC;
/* SPECIAL CHARACTER, TEST FOR VALID COMBINATION */
IF (CHARA='='1) (CHARA="("1) THEN DO; /* VALID - */
FLAG=28: RETURN; END; /* ONLY AS AN IF-STATEMENT */
IF CHARA="#" THEN DO:
CALL NEXTONE:
IF CHARA="#" THEN DO:
SYNCNT=2; SYNTAX(2)=CHARA; FLAG=6: RETURN; END;
ELSE DO:
REPEAT=1; FLAG=29: RETURN; END;
END:
IF (CHARA=">")&(CHARA="<")&(CHARA="~") THEN DG;
FLAG=29; RETURN; END; /*LOOK FOR CHARACTER PAIR */
CALL NEXTONE;
IF CHARA='=' THEN DO:
SYNTAX=2; SYNTAX(2)=CHARA; /*VALID PAIR, SIGNAL AND RETURN */
FLAG=29; RETURN; END;
REPEAT /*INVALID, SET FLAG SO CHARA IS NOT LOST */
IF SYNTAX(1)='=' THEN DO:
FLAG=29; RETURN; END; /*~INVALID UNLESS PART OF -= */
ELSE DO:
FLAG=28; RETURN; END;
/*FIRST CHARACTER MUST BE ALPHABETIC, ELSE ERROR */
NUTSPEC: DO I=1 TO 29; /*TEST FOR ALPHABETIC FIRST CHARACTER */
IF CHARA=ALPHABET(1) THEN GO TO OKALPHA;
/*NO MATCH, SIGNAL ASSIGNMENT AND UPDATE CROSS REF TABLE */
FLAG=27; IF RECORD(1)=RECORD THEN DO:
DO I=1 TO NITEM:
IF RECORD(I)=WORD THEN DO:
RETURN; END;
END;
END: /*FIRST WORD FORMED, COMPARE IT WITH ALL KEYWORDS */
END;

NOTCOMP:
LHSFLAG,SFLAG(MATRXCT)=0; /* END OF L.H.S. OPTION */
IF SYNTAX(I)='-' THEN DO: /* PARENTHESISED EXPRESSION */
PUT FILE(OBJECT) EDIT (" " (A(2)));
CALL BRACKET;
IF ENDFLAG THEN GO TO TOSOON;
GO TO STARTMTX;
END:
IF SYNTAX(I)='-' THEN GO TO BADFORM; /*, ILLEGAL HERE */
COMPEND:
/* WRITE LHS IF NEEDED AND COMPARISON OPERATOR */
WRITE LHS:
PUT FILE (OBJECT) EDIT ((LHSIDE(I) DO I=1 TO LHNCNT))
{x(1)},32 A(1));

END:
PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
{x(1)},32 A(1));
/* GET FIRST UNIT OF R.H.S. */
CALL ENTITY;
IF ENDFLAG THEN GO TO TOSOON;
IF SYNTAX(I)='(' THEN DO: /* PARENTHESISED EXPRESSION */
PUT FILE (OBJECT) EDIT (" " (A(2)));
CALL BRACKET; IF ENDFLAG THEN GO TO TOSOON;
GO TO LATTER;
END:
IF (WORD='.')[(WORD=''' THEN') THEN GO TO BADFORM;
PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
{x(1)},32 A(1));
/* READ ON TO END OF SUB-STATEMENT */
LATTER:
CALL ENTITY; IF ENDFLAG THEN GO TO TOSOON;
IF SYNTAX(I)='(' THEN DO;
PUT FILE (OBJECT) EDIT (" " (A(2)));
CALL BRACKET; IF ENDFLAG THEN GO TO TOSOON;
GO TO LATTER; END:
IF SYNTAX(I)='*' THEN DO: /* END SUB-STATEMENT */
PUT FILE (OBJECT) EDIT (" " (A(3)));
GO TO STARTMTX; END:
IF WORD='THEN' THEN DO: /* THEN ENDS STATEMENT */
CALL ENTITY; IF ENDFLAG=0 THEN DO: /* SPURIOUS - TREAT AS */
PUT FILE (OBJECT) EDIT (" THEN") (A(5)); /* NAME */
GO TO LATTER; END:
ELSE DO: /* GENUINE END OF STATEMENT */
PUT FILE (OBJECT) EDIT (" THEN") (A(10));
RETURN; END:
END:
PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
{x(1)},32 A(1));
GO TO LATTER;

I2ST:
/* ENTRY FOR SECONDARY IF-STATEMENTS */
ENTRY:
/* RESET L.H.S. CODES FOR THIS COLUMN */
LHWORD=SWORD(RHNCNT(MATRXCT)); LHNCNT=SYNCNT(MATRXCT);
LHSFLAG,SFLAG(MATRXCT)=0;
PUT FILE (OBJECT) EDIT (" END; IF") (A(9));
GO TO EXTRAENT;
/* SET ERROR FLAG FOR INCOMPLETE STATEMENT */
TOSOON:
PUT FILE(SYSPRINT) EDIT (" *** IF STATEMENT IS INCOMPLETE*)
(SKIP, A(31));
ERROR=1; RETURN;

/* SET ERROR FLAG FOR BAD STRUCTURE */
BADFORM: CALL ENTITY; IF ENDFLAG=1 THEN GO TO BADFORM;
PUT FILE (SYSPRINT) EDIT (* *** IF-STATEMENT STRUCTURE IS ',
"ILLEGAL") (SKIP, A(31), A(7));
ERROR=1; RETURN;

END IFST;
END HEST;
PROEDURE:
/* Fred Pre-Processor, David Mendus, 11/3/67
routine to process all IF statements, primary ifs from
entry IFST, secondary ifs from 12ST */

DECLARE (ENDFLAG, FIXED BINARY(1),
SYNTAX(32), PACKED CHARACTER(1),
SYNCT, FIXED BINARY,
MATRXCT, FIXED BINARY,
ERROR, FIXED BINARY(1)) STATIC EXTERNAL,
WORD, CHARACTER(32) DEFINED SYNTAX,

(LHSFLAG, FIXED BINARY(1),
LHSIDE(32), PACKED CHARACTER(1),
STFLAG(10), FIXED BINARY(1),
STWORD(10), PACKED CHARACTER(32),
STCNT(10), FIXED BINARY,
LHCNT, FIXED BINARY) STATIC,

SPWORD, CHARACTER(12) INITIAL ('=' '>' '<' '=' '<=' '=' '>=' ) STATIC,
SPCHA(6), PACKED CHARACTER(2) DEFINED SPWORD,
ALWORD, CHARACTER(29) INITIAL
('ABCDEFHijklmnopqrsuvwxyz#@ ') STATIC,
ALPHABET(29), PACKED CHARACTER(1) DEFINED ALWORD,
LHWORD, CHARACTER(32) DEFINED LHSIDE;

/* START BY TESTING FIRST WORD FOR LEGALITY */
LHSFLAG, STFLAG(MATRXCT)=0;
PUT FILE (OBJECT) EDIT ('' IF ('') (A(5));
CALL ENTITY; /* GET NEXT UNIT */
IF ENDFLAG THEN GO TO TOSDON; /* STATEMENT CANT STOP HERE */
IF SYNTAX(1)="" THEN DO; /* PARENTHESISED EXPRESSION */
PUT FILE (OBJECT) EDIT ('' ('') (A(2));
CALL Bracket; IF ENDFLAG THEN GO TO TOSDON;
ELSE GO TO STARTNXT;
END;
DO I=1 TO 29; /* IF NOT (-ISED THEN WORD MUST HAVE - */
IF SYNTAX(I)=ALPHABET(I) THEN GO TO STORE; /*ALPHA START*/
END;
GO TO BADFORM;

/* TEST FIRST WORD FOR A VALID L.H.S. */
STORE: LHSIDE=SYNTAX; STWORD(MATRXCT)=LHWORD;
STCNT(MATRXCT)=SYNCT;
CALL ENTITY; /* NEXT UNIT MUST BE COMPARISON OPERATOR */
IF ENDFLAG THEN GO TO TOSDON;
DO I=1 TO 6;
IF WORD=SPCHA(I) THEN DO;
LHSFLAG, STFLAG(MATRXCT)=1; GO TO WRITELHS; END;
END; /* CANNOT BE L.H.S. */ OUTPUT STORED UNIT /
PUT FILE (OBJECT) EDIT ((LHSIDE(I) DO I=1 TO LHOCNT) (X(I),
A(I)));
GO TO NOTCOMP;

/* READ L.H.S. OF NEXT SUB-STATEMENT IF PRESENT */
STARTNXT: CALL ENTITY; IF ENDFLAG THEN GO TO TOSDON;
EXTRAENT: DO I=1 TO 6; /* TEST FOR COMPARISON OPERATOR */
IF WORD=SPCHA(I) THEN GO TO COMPEND;
GOST: PROEDURE:
/* FRED PRE-PROCESSOR, DAVID MENDUS, 16/3/67
TO COMPILE A GO TO STATEMENT AND EDIT THE LABEL NAME */

DECLARE ENDFLAG FIXED BINARY (1),
SYNTAX(32) PACKED CHARACTER (1),
SYNCTOT FIXED BINARY,
ERROR FIXED BINARY (1) STATIC EXTERNAL,
ALPHA CHARACTER (29) INITIAL
(*ABCDEFGHIJKLMNPOQRSTUVWXYZ#*) STATIC,
ALPHABET(29) PACKED CHARACTER (1) DEFINED ALPHA;

CALL ENTITY: /* NEXT UNIT MUST BE 'TO' */
IF ENDFLAG THEN GO TO TOOSoon;
IF (SYNCTOT=2) {
(SYNTAX(1) -= 'T') {
(SYNTAX(2) -= 'O') THEN
GO TO BADSTR;
CALL ENTITY: /* THIRD UNIT MUST BE LABEL */
IF ENDFLAG THEN GO TO TOOSoon;
IF SYNCTOT5 THEN GO TO BADLAB; /* LABEL TOO LONG */
DO I=1 TO 29; /* FIRST CHARACTER MUST BE ALPHA */
IF SYNTAX(1) = ALPHA(1) THEN GO TO WRITE; END;

BADLAB: PUT FILE (SYSPRINT) EDIT (' *** LABEL FORMAT INVALID'*)
(SKIP, A(25));
SKIPEND: CALL ENTITY; IF ENDFLAG THEN RETURN; GO TO SKIPEND;

WRITE: PUT FILE (OBJECT) EDIT (' GO TO', (SYNTAX(1) DO I=1 TO SYNCTOT
1, ' ' A(I), 6 A(11));

CALL ENTITY: IF ENDFLAG THEN RETURN; /* ERROR IF MORE */
BADSTR: PUT FILE (SYSPRINT) EDIT (' *** FORMAT OF GO TO STATEMENT',
' * INVALID '*)(SKIP, A(30), A(8));
GO TO SKIPEND;

TOOSoon: PUT FILE (SYSPRINT) EDIT (' *** GO TO STATEMENT INCOMPLETE')
(SKIP, A(31));
RETURN;

END GOST;
HEST: PROCEDURE:
/* FRED PRE-PROCESSOR, DAVID WENDUS, 26/3/67
TO EDIT AND PROCEED A HEADING STATEMENT, WRITING AN
ON ENDPAGE CONDITION ACTION, FILE MUST BE DECLARED */
DECLARE (ENDFLAG FIXED BINARY(1),
CHARA CHARACTER(1),
HEADFLG FIXED BINARY(1),
FILEFLG FIXED BINARY(1),
FILENAME(32) PACKED CHARACTER(1),
FILECNT FIXED BINARY,
ERROR FIXED BINARY(1),
INWORD FIXED BINARY(1),
) STATIC EXTERNAL,
HEADING(133) PACKED CHARACTER(1),
HEADCNT FIXED BINARY;
/* THERE MAY ONLY BE ONE HEADING STATEMENT */
IF HEADFLG THEN DO:
PUT FILE (SYSPRINT) EDIT ("*** THIS IS NOT THE FIRST ",
'HEADING STATEMENT') (SKIP,A(27),A(117)); ERROR=1;
GO TO BADEND; END;
HEADFLG=1; /* FIRST SIGNIFICANT CHARACTER SHOULD BE */
CALL NEXTONE; IF ENDFLAG THEN DO:
TOOOSON:
PUT FILE (SYSPRINT) EDIT ("*** HEADING STATEMENT IS ",
INCOMPLETE") (SKIP,A(26),A(10)); ERROR=1; RETURN;
END:
BADFORM:
PUT FILE (SYSPRINT) EDIT ("*** HEADING STATEMENT FORMAT ",
IS ILLEGAL") (SKIP,A(30),A(10)); ERROR=1; GO TO BADEND;
END:
HEADCNT=0; /* LOOP STORING CHARACTERS OF STRING */
LOOP:
CALL NEXTONE; IF ENDFLAG THEN GO TO TOOOSON;
IF CHARA="""" THEN GO TO ENDSTRING;
HEADCNT=HEADCNT+1;
IF HEADCNT>133 THEN DO:
PUT FILE (SYSPRINT) EDIT ("*** HEADING EXCEEDS 133 ",
CHARACTERS") (SKIP,A(25),A(10)); ERROR=1; GO TO BADEND;
END:
HEADING(HEADCNT)=CHARA; GO TO LOOP;
ENDSTRING:
CALL NEXTONE; IF ENDFLAG THEN GO TO ENDSFA;
IF INWORD="""" THEN GO TO ENDSFA; /* INTERVENING BLANK */
IF CHARA="""" THEN GO TO LOOP; /* START NEXT STRING */
ELSE GO TO BADFORM;
/* END OF STRING, COULD BE ANOTHER STRING ON END */
ENDSFA:
IF HEADCNT<133 THEN DO: /* BLANK FILL ARRAY */
DO HEADCNT=HEADCNT TO 133; HEADING(1)="""" END;
PUT FILE (OBJECT) EDIT ("ON ENDPAGE (") (A(12));
IF FILEFLG THEN PUT FILE (OBJECT) EDIT (FILENAME(1) DO I=1 TO 
FILECNT) (32 A(1));
ELSE PUT FILE (OBJECT) EDIT ("SYSPRINT") (A(8));
PUT FILE (OBJECT) EDIT ("("A(12));
IF FILEFLG THEN PUT FILE (OBJECT) EDIT (FILENAME(1) DO I=1 TO 
FILECNT) (32 A(1));
ELSE PUT FILE (OBJECT) EDIT ("SYSPRINT") (A(9));
PUT FILE (OBJECT) EDIT ("**") EDIT ("**") (HEADING(1) DO I=1 TO 
133,"""" (PAGE,A(133));") (A(9),133 A(1),A(17));
RETURN;
BADEND:
CALL NEXTONE; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;

7656
END;

/* AN EDIT BLOCK HAS BEEN FOUND, CLEAR FLAGS, READ ON. */
EXECFLAG=0; EDITFLAG=0; FLAGFLG=0; LISTFLG=0;
PAALFLG=0; PRAALFLG=0; HEADFLAG=0; MAXVAL=2;
FILEX='SYSPRINT'; FILECNT=8; SUBFLG=0;
COLONE=0; COLCNT=73; /* START VALUES FOR NEXTONE */
NEXTSTA;
GETKEYWORD; /* GET KEYWORD */
IF EXECFLG THEN DO; /*BRANCH FOR EXECUTABLE/DECLARATIVE*/
IF (FLAG<=14)&&(FLAG>26) THEN GO TO PROCEDURE;
IF FLAG=26 THEN GO TO PROCEDURE;
IF FLAG=15 THEN DO; /*END STATEMENT, TEST FOR END */
CALL COLCHK;
/* EDIT OR SUBEDIT */
IF WORD='EDIT' THEN DO; /*END EDIT, PROCESS & CONTINUE */
CALL COMPLT; GO TO LOCKNEXT; END;
IF WORD='SUBEDIT' THEN DO; /*END SUBEDIT, PROCESS & CONT */
CALL ENDSUB; GO TO NEXTSTA; END;
PUT FILE(SYSPRINT) EDIT ('*** FORMAT OF END STATEMENT',
'INVALID, END EDIT STATEMENT ASSUMED') (SKIP,A(29),A(35));
ERROR=1;
CALL COMPLT; GO TO LOCKNEXT;
END;

/* DECLARATIVE STATEMENT FOLLOWS EXECUTABLE STATEMENT(S) */
PUT FILE(SYSPRINT) EDIT ('*** ALL DECLARATIVE STATEMENTS',
'MUST PRECEDE THE FIRST EXECUTABLE STATEMENT')
(Skip,A(32),A(48)); ERROR=1;
FLAG=29; GO TO PROCESS;
END;

/* EXECUTABLE FLAG NOT SET, EXECUTABLE STATEMENT POSSIBLE */
ELSE DO;
IF (FLAG>=14)&&(FLAG<=25) && (FLAG=29) THEN GO TO PROCESS;
IF (EDITFLAG=0)&&(FLAG=0)&&(LISTFLG=0) THEN DO;
/* DECLARATIVE STATEMENTS NOT COMPLETE */
PUT FILE(SYSPRINT) EDIT ('** THE DECLARATIVE SECTION',
'MUST CONTAIN ONE EDIT, ONE FLAG AND ONE LIST OR',
'*PRINTLINE STATEMENT') (SKIP,A(29),A(48),A(19));
ERROR=1;
END;
EXECFLAG=1; CALL EXECONE;
IF FLAG=26 THEN GO TO PROCESS;
END;

/* CALL SUBROUTINES TO CHECK COLUMN AND PROCESS STATEMENT */
PROCEDURE CALL COLCHK;
PROCESS: IF FLAG=1 THEN DO; CALL IFST; GO TO EX; END;
IF FLAG= 2 THEN DO; CALL ERST; GO TO EX; END;
IF FLAG= 3 THEN DO; CALL QUST; GO TO EX; END;
IF FLAG= 4 THEN DO; CALL ADST; GO TO EX; END;
IF FLAG= 5 THEN DO; CALL ELST; GO TO EX; END;
IF FLAG= 6 THEN DO; CALL CST; GO TO EX; END;
IF FLAG= 7 THEN DO; CALL LST; GO TO EX; END;
IF FLAG= 8 THEN DO; CALL DOST; GO TO EX; END;
IF FLAG=11 THEN DO; CALL DOST; GO TO EX; END;
IF FLAG=12 THEN DO; CALL CHST; GO TO EX; END;
IF FLAG=13 THEN DO; CALL PEST; GO TO EX; END;
IF FLAG=14 THEN DO; CALL STST; GO TO EX; END;
IF FLAG=16 THEN DO; CALL DCST; GO TO EX; END;
IF FLAG=17 THEN DO: CALL EDST; GO TO EX; END;
IF FLAG=18 THEN DO: CALL FLST; GO TO EX; END;
IF FLAG=19 THEN DO: CALL MAST; GO TO EX; END;
IF FLAG=20 THEN DO: CALL LIST; GO TO EX; END;
IF FLAG=21 THEN DO: CALL PNST; GO TO EX; END;
IF FLAG=22 THEN DO: CALL POST; GO TO EX; END;
IF FLAG=23 THEN DO: CALL PST1; GO TO EX; END;
IF FLAG=24 THEN DO: CALL HEST; GO TO EX; END;
IF FLAG=25 THEN DO: CALL PRST; GO TO EX; END;
IF FLAG=26 THEN DO: CALL LAST; GO TO EX; END;
IF FLAG=27 THEN DO: CALL AIST; GO TO EX; END;
IF FLAG=28 THEN DO: CALL 12ST; GO TO EX; END;
IF FLAG=29 THEN DO: CALL XXST; GO TO EX; END;
/ * IF IT WAS EXECUTABLE THEN SET LIST OF XREF FLAGS /*
EX: IF ALLOCATION(RECORD) THEN DO:
IF (FLAG<13)&&(FLAG=27))&&(FLAG=28) THEN DO:
DO I=1 TO NITEM:
IF RECFLAG(I)='1'B THEN DO:
DO J=1 TO NITEM:
IF RECFLAG(J)='1'B THEN XREF(I,J)='1'B;
END:
END:
END:
DO I=1 TO NITEM: RECFLAG(I)='0'B; END:
END:
GO TO NEXTSTA;
END FRED:
PROCEDURE:
/* FRED PRE-PROCESSOR, DAVID MENDUS, 12/3/67
TO EDIT AND PROCESS THE FLAG STATEMENT */

DECLARE (ENDFLAG FIXED BINARY(1),
ERROR FIXED BINARY(1),
SYNTAX FIXED BINARY,
SYNCNT FIXED BINARY,
FLGORDO FIXED BINARY,
FLGCTN FIXED BINARY,
FLAGFLG FIXED BINARY(1) STATIC EXTERNAL,
ALPHA CHARACTER(29) INITIAL
('ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789@#') STATIC,
ALPHABET(29) PACKED CHARACTER(1) DEFINED ALPHA;

;/* THERE MUST BE ONE AND ONLY ONE FLAG STATEMENT */
IF FLAGFLG=1 THEN DO: /* PREVIOUS STATEMENT */
/* PUT FILE (SYSPRINT) EDIT ('*** THIS IS NOT THE FIRST FLAG',
STATEMENT) (SKIP,A(31),A(101));
ERROR=1; GO TO BADEND;*/
END:
FLAGFLG=1; /* SET THE FLAG */
CALL ENTITY:
IF ENDFLAG=1 THEN DO: /* STATEMENT CANNOT STOP HERE */
/* PUT FILE (SYSPRINT) EDIT ('*** NO FLAG IS SPECIFIED')
(SKIP,A(25));
ERROR=1; RETURN; END;*/
DO I=1 TO 29: /* FIRST CHARA MUST BE ALPHA */
IF SYNTAX(I)=ALPHABET(I) THEN GO TO VALID;
END:
/* PUT FILE (SYSPRINT) EDIT ('*** FLAG NAME MUST HAVE ',
'ALPHABETIC FIRST CHARACTER') (SKIP,A(25),A(26));
ERROR=1; GO TO BADEND;*/

VALID: FLMOR=SYNTAX; FLGCTN=SYNCNT; /*STORE WORD */
CALL ENTITY: /* THIS SHOULD BE END OF STATEMENT */
IF ENDFLAG=1 THEN RETURN;
/* PUT FILE (SYSPRINT) EDIT ('*** FLAG NAME IS NOT SINGLE WORD',
' FORMAT') (SKIP,A(33),A(7));
ERROR=1;*/
BADEND: CALL ENTITY; IF ENDFLAG=1 THEN GO TO BADEND;
RETURN;
END FLST:
PROCEDURE: /* FRED PEE-PROCESSOR, DAVID MENDUS, 26/3/67 TO EDIT AND PROCESS A FORMAT STATEMENT */

DECLARE (ENDFLAG FIXED BINARY(1),
SYNTAX(32) PACKED CHARACTER(1),
SYNTAX(1) FIXED BINARY,
ERROR FIXED BINARY(1),
FORFLAG FIXED BINARY(1)) STATIC EXTERNAL,
BRCNT FIXED BINARY;

/* SHOULD BE ONLY ONE FORMAT STATEMENT */
IF FORFLAG THEN DO:
PUT FILE (SYSPRINT) E IT (' *** THIS IS NOT THE FIRST ',
'FORMAT STATEMENT') (SKIP, A(27), A(16)); ERROR=1;
GO TO BADEND; END;

FORFLAG=1; /* WRITE LABEL AND FORMAT */
PUT FILE (OBJECT) EDIT (' Q #FORM: FORMAT ') (A(17));
CALL ENTITY; IF ENDFLAG THEN DO:
PUT FILE (SYSPRINT) EDIT (' *** FORMAT STATEMENT IS ',
'INCOMPLETE') (SKIP, A(25), A(10)); ERROR=1; RETURN;
END;
IF SYNTAX(1)='(' THEN DO:
PUT FILE (SYSPRINT) EDIT (' *** FORMAT DOES NOT START ',
WITH ' )') (SKIP, A(27), A(6)); ERROR=1; GO TO BADEND;
END;
BRCNT=1; /* COUNT OF EXCESS ( OVER ) */
LOOP:
PUT FILE (OBJECT) EDIT ((SYNTAX(1) DO I=1 TO SYNTAXNT))
(X(1), 32 A(1));
CALL ENTITY; IF ENDFLAG THEN DO:
IF BRCNT=0 THEN GO TO CLOSE; ELSE GO TO TOOSON; END;
IF SYNTAX(1)='(' THEN DO:
BRCNT=BRCNT+1; GO TO LOOP; END;
IF SYNTAX(1)=')' THEN DO:
BRCNT=BRCNT-1; IF BRCNT<0 THEN DO:
PUT FILE (SYSPRINT) EDIT (' *** RIGHT PARENTHESES EXCEED',
' LEFT PARENTHESES ') (SKIP, A(29), A(17)); ERROR=1;
BADEND:
CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
END;
GO TO LOOP; END;
GO TO LOOP;
CLOSE:
PUT FILE (OBJECT) EDIT (";") (A(1));
END POST:
DECLARE (ERROR FIXED BINARY(1),
COLOM FIXED BINARY,
COLECT FIXED BINARY,
ENDPROG FIXED BINARY(1),
EDITFLG FIXED BINARY(1),
FLAGFLG FIXED BINARY(1),
PORFLG FIXED BINARY(1),
FILEFLG FIXED BINARY(1),
FILENAME(32) PACKED CHARACTER(1),
FILECNT FIXED BINARY,
PEALFLG FIXED BINARY(1),
HEADFLG FIXED BINARY(1),
MAXVAL FIXED BINARY,
LISTFLG FIXED BINARY(1),
SUBFLG FIXED BINARY(1),
FLAG FIXED BINARY,
SYNTAX(32) PACKED CHARACTER(1),
ITEM FIXED BINARY ) STATIC EXTERNAL,
WORD CHARACTER(32) DEFINED SYNTAX,
(RECFLAG(ITEM) BIT(1),
RECORD(ITEM) PACKED CHARACTER(12),
XREF(ITEM,ITEM) BIT(1) CONTROLLED EXTERNAL,
FILEXY CHARACTER(32) DEFINED FILENAME,
EXECFLG FIXED BINARY(1) STATIC;

/* PAGE HEADINGS */
ON ENDPAGE (SYSPRINT) BEGIN;
PAGE=1;
PUT FILE (SYSPRINT) EDIT ("FRED PRE-PROCESSOR ***", "PAGE", [PAGE]) (PAGE,A(23),X(89),A(4),F(4,0));
PUT FILE (SYSPRINT) SKIP;
END;
PAGE=1;
PUT FILE (SYSPRINT) EDIT ("FRED PRE-PROCESSOR ***", "PAGE", [PAGE]) (PAGE,A(23),X(89),A(4),F(4,0));
PUT FILE (SYSPRINT) SKIP;
/* READ THROUGH DECK UNTIL START EDIT STATEMENT, PROCESS IT. */
ERROR=0; ENDPROC=0;
LOOKNEXT=CALL STARTED; /* TO LOOK FOR NEXT START EDIT STATEMENT*/
IF ENDPROC THEN DO;
IF ERROR THEN DO;
PUT FILE(SYSPRINT) EDIT ("END OF THIS COMPIpATION", ",", "ERRORS WERE NOTED AS SIGNALLED ABOVE", ",", "DO NOT", "ATTEMPT TO RUN THIS PROGRAM UNTIL THESE ERRORS HAVE", "BEEN REMOVED," ) (SKIP(3),A(26),A(39),SKIP,A(8),A(52),
A(13));
RETURN; END; /*STOP' ATER TESTING */
ELSE DO;
PUT FILE(SYSPRINT) EDIT ("END OF THIS COMPIpATION", ",", "NO ERRORS WERE NOTED IN THE EDITING STATEMENTS") (SKIP(3),A(26),A(46));
RETURN; END; /*STOP' ATER TESTING */
FIST: PROCEDURE: /* FRED PREE-PROCESSOR, DAVID MENOUS, 24/3/67 TO EDIT AND PROCESS A FILE STATEMENT */

DECLARE (ENDFLAG FIXED BINARY(1),
SYNTAX(32) PACKED CHARACTER(1),
SYNCT FIXED BINARY,
ERROR FIXED BINARY(1),
HEADFLAG FIXED BINARY(1),
FILENAME(32) PACKED CHARACTER(1),
FILECNT FIXED BINARY ) STATIC EXTERNAL,

ALPHA CHARACTER(29) STATIC INITIAL
('ABCDEFGHIJKLMNOPQRSTUVWXYZ#~$&'),

IF HEADFLAG THEN DO: /* MUST PRECEDE HEADING STATEMENT */,
  PUT FILE (SYSPRINT) EDIT ('*** THIS DOES NOT PRECEDE ',
    'THE HEADING STATEMENT') (SKIP,A(27),A(21));
  ERROR=1: GO TO BADEND: END;

IF FILEFLAG THEN DO: /* ONLY ONE FILE STATEMENT ALLOWED */,
  PUT FILE (SYSPRINT) EDIT ('*** THIS IS NOT THE FIRST FILE',
    ' STATEMENT') (SKIP,A(31),A(101)); ERROR=1;
  CALL ENTITY: IF ENDFLAG=0 THEN GO TO BADEND; RETURN;

BADEND: END;

FILEFLG=1: /* SET FLAG, READ FILE NAME */
CALL ENTITY;
IF ENDFLAG THEN DO;
  PUT FILE (SYSPRINT) EDIT ('*** FILE STATEMENT IS ',
    'INCOMPLETE') (SKIP,A(23),A(101)); ERROR=1: RETURN;

/* CHECK FIRST CHARACTER FOR ALPHA AND STORE NAME */
DO I=1 TO 29; IF SYNTAX(I)=ALPHABET(I) THEN GO TO OK; END;
  PUT FILE (SYSPRINT) EDIT ('*** FILE NAME GIVEN IS INVALID'),
    (SKIP,A(31)); ERROR=1: GO TO BADEND;

OK: FILENAME=SYNTAX; FILECNT=SYNCT: /* SHOULD BE NO MORE */
CALL ENTITY: IF ENDFLAG THEN RETURN;
  PUT FILE (SYSPRINT) EDIT ('*** FILE STATEMENT FORMAT INVALID' ),
    (SKIP,A(34)); ERROR=1: GO TO BADEND;

END FIST;
NEXTRHS: CALL ENTITY; IF ENDFLAG THEN GO TO CHECKEND;
  PUT FILE (OBJECT) EDIT ((STORE(I) DO I=1 TO STRCNT))
     (X(1),32 A(1));
  STORE=SYNTAX; STRCNT=SYNCNT; GO TO NEXTRHS;
CHECKEND: IF (STORE(1)="**") (SYNCNT=1) THEN DO:
  PUT FILE (OBJECT) EDIT (** **A29)); RETURN; END;
  PUT FILE (SYSPRINT) EDIT (' *** ILLEGAL END TO STATEMENT')
     (A29));
ERROR=1; RETURN;
FINISH: CALL ENTITY; IF ENDFLAG THEN RETURN;
  PUT FILE (SYSPRINT) EDIT (' *** ILLEGAL END TO STATEMENT')
     (A29));
BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
END ERST:
EXECONE: PROCEDURE;
/* FRED PRE-PROCESSOR, DAVID MENDUS, 27/3/67
   TO WRITE FIRST EXECUTABLE STATEMENTS OF OBJECT CODE */
DECLARE NITEM FIXED BINARY STATIC EXTERNAL;
   RECORD(NITEM) PACKED CHARACTER(12) CONTROLLED EXTERNAL;
   PUT FILE (OBJECT) EDIT ('Q.#FATAL, Q.#QUERY, Q.#ADJUST,';
   'Q.#DISC=0:') (A(28), A(11));
   IF -ALLOCATION(RECORD) THEN RETURN;
   DO I=1 TO NITEM;
      PUT FILE (OBJECT) EDIT ('Q.#', RECORD(I), ' = ', RECORD(I), ' ; ')
         (A(3), A(12), A(11), A(12), A(11));
   END;
   RETURN;
END EXECONE:
ERST: PROCEDURE: /* ENTRY QU5T; ADCON */
/* FRED PRE-PROCESSOR; DAVID MENDUS, 18/3/67 */
TO PROCESS ERROR OR QUERY STATEMENTS INCLUDING ERROR FLAG
SETTING AND DIAGNOSTIC ACTION. ALSO COMPLETES ADJUST
STATEMENT DECODING */

DECLARE (ENDFLAG FIXED BINARY (1),
SYNTAX(32) PACKED CHARACTER (1),
SYNCT FIXED BINARY,
ERROR FIXED BINARY(1),
LISTFLG FIXED BINARY(1) ) STATIC EXTERNAL,
(STORE(32) PACKED CHARACTER(1),
STRNCT FIXED BINARY ) STATIC:

/* WRITE CODE TO SET ERROR FLAG */
PUT FILE (OBJECT) EDIT ("_O_#FATAL=1:*1") (A(12)); GO TO READON;

/* WRITE CODE TO SET QUERY FLAG */
PUT FILE (OBJECT) EDIT ("_O_#QUERY=1:*1") (A(12));
ADCON: ENTRY: /* TO COMPLETE DECODING ADJUST STATEMENT */

READON: CALL ENTITY: /* NEXT UNIT SELECTS DIAGNOSTIC OPTION */
ENDSONN: PUT FILE (SYSPRINT) EDIT ("### STATEMENT INCOMPLETE")
(SKIP,A(25)); ERROR=1; RETURN; END;

IF (SYNCT=1)OR(SYNTAX(1)=1) THEN GO TO FLAGIT;
IF LISTFLG=-1 THEN DO: /* ILLEGAL LISTING OPTION */
PUT FILE (SYSPRINT) EDIT ("### ERROR MESSAGE IS ILLEGAL -
AS PRINTNONE OPTION IS SELECTED") (SKIP,A(30),A(31));
ERROR=1; GO TO BADEND; END;
IF (SYNTAX(1)=") THEN DO: /* IF O #DIGCT=40 THEN O #DIGCT=
"O #DIGCT=") (A(30),A(12));
IF SYNTAX(1)="**** THEN DO; /* CHARACTER STRING CONSTANT */
PUT FILE (OBJECT) EDIT ("_O_#DIGCTO.#DIGCT=") (SYNTAX(1))
DO I=1 TO SYNTAX(1),I); (A(19),33 A(1)); END;
ELSE DO; /* CHARACTER STRING VARIABLE */
PUT FILE (OBJECT) EDIT ("_O_#DIGCTO.#DIGCT=") (SYNTAX(1))
DO I=1 TO SYNTAX(1),I); (A(19),33 A(1)); END;
GO TO FINISH;

FLAGIT: IF LISTFLG=1 THEN DO: /* MUST BE AN ERROR MESSAGE */
PUT FILE (SYSPRINT) EDIT ("### AN ERROR MESSAGE MUST BE -
'GIVEN AS LISTING OPTION WAS SELECTED'") (SKIP,A(30),A(36));
ERROR=1; GO TO BADEND; END;

READHS: CALL ENTITY: IF ENDFLAG THEN GO TO ENDSONN:
WRITETHS:PUT FILE (OBJECT) EDIT (SYNTAX(1) DO I=1 TO SYNCT)
(X(1),32 A(1))); /* FIRST PART OF L.H.S. */
WRITEQ: CALL ENTITY: IF ENDFLAG THEN GO TO ENDSONN:
IF SYNTAX(1)="=" THEN GO TO WRITETHS;
IF SYNTAX(1)="=" THEN DO; /* BRACKET VALID HERE */
P ut FILE (OBJECT) EDIT ("(") (X(1),A(1));
CALL BRACKET; GO TO READHS; END;
PUT FILE (SYSPRINT) EDIT ("### INVALID FORM OF ASSIGNMENT")
(SKIP,A(31)); ERROR=1; GO TO BADEND;
WRITETHS:PUT FILE (OBJECT) EDIT ("=") (X(1),A(1)); STORE="1";}
RETURN;
END;
ELSE DO:
/* ANOTHER CHARACTER, CHECK VALID LENGTH */
IF SYNCNT<32 THEN DO:
/* STORE NEXT CHARACTER */
SYNCNT=SYNCNT+1; SYNTAX(SYNCNT)=CHARA;
GO TO ALPHA; END;
ELSE DO:
/* TOO LONG */
PUT FILE (SYSPRINT) EDIT ('*** IDENTIFIER EXCEEDS',
'32 CHARACTERS?') (ERROR=1)
DO WHILE (INWORD=1):
/* SKIP CHARACTERS TO NEXT BLANK */
CALL NEXTONE; END;
RETURN;
END;
END ENTITY;
ENTITY:  PROCEDURE:  
/* FRED PRE-PROCESSOR, DAVID MENDUS, 20/2/67
Routines called as part of the Recogniser to assemble the
next word or string of special characters. Updates the
CROSS REFERENCE TABLE if appropriate. */

DECLARE (ENDFLAG  FIXED BINARY(1),
INWORD  FIXED BINARY(1),
REPEAT  FIXED BINARY(1),
ERROR  FIXED BINARY(1),
CHARA  CHARACTER(1),
SYNTAX(32) PACKED CHARACTER(1),
SYNCT  FIXED BINARY STATIC EXTERNAL,
WORD  CHARACTER(32) DEFINED SYNTAX,
(RECORD(NITEM) PACKED CHARACTER(12),
RECEFlag(NITEM) BIT(1)  CONTROLLED EXTERNAL,
NITEM  FIXED BINARY STATIC EXTERNAL;
/* GET FIRST CHARACTER, IF END OF STATEMENT THEN RETURN */

WORD=';

RESTART:  CALL NEXTONE;
IF ENDFLAG THEN RETURN;
IF INWORD=0 THEN GO TO RESTART;
SYNCT=1; SYNTAX(1)=CHARA;
IF INWORD=-1 THEN GO TO ALPHA;
/* SPECIAL CHARACTER, RETURN CHARACTER OR PAIR */
IF CHARA='*'(CHARA='<') THEN DO:
   CALL NEXTONE;
   /* >=,=<,=< ARE VALID */
ELSE DO:
   REPEAT=1; RETURN: END;
END;
IF CHARA='*' THEN DO:
   /* ** IS VALID */
   CALL NEXTONE;
   IF CHARA='*' THEN DO:
   SYNCT=2; SYNTAX(2)=CHARA; RETURN: END;
ELSE DO:
   REPEAT=1; RETURN: END;
END;
IF CHARA='1' THEN DO:
   /* || IS VALID */
   CALL NEXTONE;
   IF CHARA='1' THEN DO:
   SYNCT=2; SYNTAX(2)=CHARA; RETURN: END;
ELSE DO:
   REPEAT=1; RETURN: END;
END:
RETURN;
/* ALPHA CHARACTER, BUILD UP STRING OF CHARACTERS */

ALPHA:
CALL NEXTONE;
IF INWORD=0 THEN DO;
/* END OF CHARACTER STRING */
IF ALLOCATION(RECORD) THEN DO;
   DO I=1 TO NITEM:
      /*LOOK FOR THE NAME OF AN ITEM */
   IF RECORD(I)=WORD THEN DO:
      RECEFlag(I)=1; RETURN: END;
END;
END;
ENDSUB:  PROCEDURE:
  /* FRED PRE/PROCESSOR, DAVID MENDUS, 24/3/67
  TO PROCESS AN END SUBEDIT STATEMENT */

DECLARE (ENDFLAG FIXED BINARY(1),
  SYNTAX(32) PACKED CHARACTER(1),
  SYMEXT FIXED BINARY,
  ERROR FIXED BINARY(1),
  SUBCNT FIXED BINARY(1),
  SUBNAME(32) PACKED CHARACTER(1),
  SUBNAME (FIXED BINARY ) STATIC EXTERNAL,
  WORD CHARACTER(32) DEFINED SYNTAX,
  WORDX CHARACTER(32) DEFINED SUBNAME;

IF SUBFLG=0 THEN DO: /* OUTSIDE SUBEDIT, ERROR */
  PUT FILE (SYSPRINT) EDIT ("*** THIS STATEMENT DOES NOT ",
  " OCCUR IN A SUBEDIT") (SKIP, A(29), A(18)); ERROR=1;
  GO TO BADEND; END;
SUBFLG=0: PUT FILE (OBJECT) EDIT (" END") (A(4));

CALL ENTITY; /* SUBEDIT MAY OR MAY NOT BE NAMED */
IF ENDFLAG=0 THEN DO: /* CHECK NAME IF GIVEN */
  IF WORD=WORDX THEN DO:
      PUT FILE (SYSPRINT) EDIT (" *** NAME GIVEN IN END ",
      " STATEMENT IS NOT NAME OF SUBEDIT") (SKIP, A(23), A(32));
      ERROR=1; GO TO BADEND;
  END:
  PUT FILE (OBJECT) EDIT (" SUBNAME(1) DO =1 TO SUBCNT ", '');
  CALL ENTITY; /* SHOULD BE END OF STATEMENT */
  IF ENDFLAG THEN RETURN;
PUT FILE (SYSPRINT) EDIT (" *** END SUBEDIT FORMAT INVALID")
  (SKIP, A(31)); ERROR=1;
BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;

END ENDSUB:
PROCEDURE:  
/* FRED PRE-PROCESSOR, DAVID MENDUS, 12/3/67  
TO PROCESS AN ELSE-STATEMENT, THE IF-BLOCKING HAS  
ALREADY BEEN DONE BY COLCHK. */

DECLARE (ENDFLAG FIXED BINARY (1)), (ERROR FIXED BINARY (1)) STATIC EXTERNAL;

CALL ENTITY; IF ENDFLAG=1 THEN GO TO WRITE;
PUT FILE (SYSPRINT) EDIT ('*** ELSE STATEMENT CONTAINS ', 'ILLEGAL WORDS') (SKIP, A(29), A(13));
ERROR=1;

READ:  CALL ENTITY; IF ENDFLAG=1 THEN GO TO READ;

WRITE:  PUT FILE (OBJECT) EDIT (' END:') (A(5));

END ELST:
RETURN;
ALLOCATE RECORD; ALLOCATE RECFLAG; ALLOCATE XREF:
DO I=1 TO NITEM; RECORD(I)=RECARRAY(I); END;
  DO I=1 TO NITEM; RECFLAG(I)='O'B;
  DO J=1 TO NITEM; XREF(I,J)='O'B; END;
  END;
  RETURN;
BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
END EDT:
PROCEDURE:  
/* FRED PRE-PROCESSOR, DAVID MENDUS, 26/3/67  
TO EDIT AND PROCESS THE EDIT STATEMENT. WRITES DECLARE  
STATEMENT FOR COPY AND LISTS ITEM NAMES FOR EDITS AND  
X-REF TABLES. */

DECLARE (ENDFLAG FIXED BINARY(1),  
SYNTAX(32) PACKED CHARACTER(11),  
SYNCNT FIXED BINARY,  
ERROR FIXED BINARY(1),  
EDITFLG FIXED BINARY(1),  
NITEM FIXED BINARY ) STATIC EXTERNAL,  
WORD CHARACTER(32) DEFINED SYNTAX,  
(RECFLAG(NITEM) BIT(1),  
RECORD(NITEM) PACKED CHARACTER(12),  
XREF(NITEM,NITEM) BIT(1) ) CONTROLLED EXTERNAL,  
RECCARRAY(100) PACKED CHARACTER(12),  
ALPHA CHARACTER(29) STATIC INITIAL  
('ABCDEFGHJKLMNOPQRSTUVWXYZ'),  
ALPHABET(29) PACKED CHARACTER(1) DEFINED ALPHA,  
NUMBERS(10) PACKED CHARACTER(1) DEFINED NUMB;

IF EDITFLG THEN DO: /* CAN ONLY BE ONE EDIT STATEMENT */  
PUT FILE (SYSPRINT) EDIT ('** THIS IS NOT THE FIRST',  
'EDIT STATEMENT') (SKIP,A(27),A(14)); ERROR=1;  
GO TO BADEND; END;

EDITFLG=1; NITEM=0;  
PUT FILE (OBJECT) EDIT (' DECLARE') (A(8));

LOOP1:  
CALL ENTITY; IF ENDFLAG THEN DO:  
PUT FILE (SYSPRINT) EDIT ('** EDIT STATEMENT IS',  
'INCOMPLETE') (SKIP,A(23),A(10)); ERROR=1; RETURN;  
END: /* CHECK FIRST CHAR FOR ALPHA. IF SO ASSUME NAME */  
DO I=1 TO 29: IF SYNTAX(1)=ALPHABET(1) THEN GO TO NAME; END;

/* IF NUMERIC MAY BE LEVEL WRITE AND LOOK FOR NAME */  
DO I=1 TO 10: IF SYNTAX(1)=NUMBERS(1) THEN DO:  
PUT FILE (OBJECT) EDIT (SYNTAX(1) DO I=1 TO SYNCNT)  
(WORD); END; END;

GO TO WRITEIT; /* NOT NAME NOR NUMBER */

NAME:  
IF SYNCNT=12 THEN DO:  
PUT FILE (SYSPRINT) EDIT ('** ITEM NAME EXCEEDS 12',  
'CHARACTERS') (SKIP,A(27),A(10)); ERROR=1; GO TO BADEND;

END:  
NITEM=NITEM+1; RECCARRAY(NITEM)=WORD; /* STORE NAME */  
PUT FILE (OBJECT) EDIT ('Q_#',SYNTAX(1) DO I=1 TO SYNCNT)  
(A(4),32 A(11));

/* NEXT UNIT CANNOT BE ITEM NAME */

LOOP2:  
CALL ENTITY; IF ENDFLAG THEN GO TO TIDYUP;  
WRITEIT: PUT FILE (OBJECT) EDIT (SYNTAX(1) DO I=1 TO SYNCNT)  
(WORD); OR (THEN NEXT MAY BE NAME */  
IF (SYNTAX(1)=1)SYNTAX(1)="" THEN GO TO LOOP1;  
ELSE GO TO LOOP2;

/* ALL DONE, TIDY UP AND ALLOCATE ARRAYS */

TIDYUP: PUT FILE (OBJECT) EDIT (''); (A(11));
PROCEDURE: /∗ FRED PRE-PROCESSOR, DAVID MENDUS, 9/3/67
TO PROCESS AN EMBEDDED DO GROUP. ∗/

DECLARE (ENDFLAG FIXED BINARY(1),
SYNTAX(32) PACKED CHARACTER(1),
SYNCNT FIXED BINARY,
ERROR FIXED BINARY(1) ) STATIC EXTERNAL,
(ONEFLAG FIXED BINARY(1),
BUFFER(32) PACKED CHARACTER(1),
BUFCT FIXED BINARY ) STATIC;

DO I=1 TO 32; BUFFER(I)=' '; END;
ONEFLAG=1;

NEXTUNIT: CALL ENTITY; /∗ GET NEXT UNIT ∗/
IF ENDFLAG THEN DO; /∗ IF ENDFLAG SET, THEN LAST ∗/
IF (BUFFER(I)=’E’); /∗ UNIT MUST BE END, ELSE ERROR ∗/
(BUFFER(2)=’N’)(BUFFER(3)=’D’)(BUFCT=’X’,32) THEN RETURN;
PUT FILE (SYSPRINT) EDIT (’ *** THIS DO-STATEMENT DOES’,
’ NOT HAVE VALID END’) (SYSPRINT,A(27), A(18));
ERROR=1; RETURN;
END;

IF ONEFLAG THEN DO; /∗ NO OUTPUT IF ONEFLAG IS SET ∗/
ONEFLAG=0; END;
ELSE DO;
PUT FILE (OBJECT) EDIT ((BUFFER(I) DO I=1 TO BUFCT))
(X(1),32 A(11)); END;
BUFCT=SYNTAX; BUFCT=SYNCNT; /∗ STORE THIS UNIT FOR NEXT ∗/
GO TO NEXTUNIT;

END DOST:
IF BRSTC THEN DO:
   PUT FILE (OBJECT) EDIT ((BRSTC(I) DO I=1 TO BRSTC))
   (X(I),64 A(I)): END;
PUT FILE (OBJECT) EDIT (',Q_#',(WORDSTC(I) DO I=1 TO WORDSTC))
   (A(4),32 A(I));
IF BRSTC THEN DO:
   PUT FILE (OBJECT) EDIT ((BRSTC(I) DO I=1 TO BRSTC))
   (X(I),64 A(I)): END;
PUT FILE (OBJECT) EDIT ('=' (A(I)));
/* PROCESS R.H.S. */
CALL ENTITY; IF ENDFLAG=0 THEN GO TO WRITERHS;
TODSOON: PUT FILE (SYSPRINT) EDIT ( ' *** THIS STATEMENT IS INCOMPLETE' )
   (SKIP,A(33));
ERROR=1; RETURN;
WRITERHS:PUT FILE (OBJECT) EDIT ((SYNTAX(I) DO I=1 TO SYNCNT))
   (X(I),32 A(I));
   CALL ENTITY; IF ENDFLAG=0 THEN GO TO WRITERHS;
   PUT FILE (OBJECT) EDIT (' ;' (A(2));
RETURN;
BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
END BRST;
PROCEDURE:
/* FRED PRE-PROCESSOR, DAVID MENDUS, 19/3/67
   EDIT AND PROCESS A DEFINE OR CHANGE STATEMENT. ACTION
   IS IDENTICAL FOR BOTH */

DECLARE (ENDFLAG FIXED BINARY (1),
SYNTAX(32) PACKED CHARACTER (1),
SYNCT FIXED BINARY,
ERROR FIXED BINARY (1),
NITEM FIXED BINARY ) STATIC EXTERNAL,
RECORD(IITEM) PACKED CHARACTER (12) CONTROLLED EXTERNAL,
WORD CHARACTER (32) DEFINED SYNTAX,
(BRCNT FIXED BINARY,
WORDST(32) PACKED CHARACTER (1),
BRST(64) PACKED CHARACTER (1),
WORDSTC FIXED BINARY,
BRSTC FIXED BINARY ) STATIC;

/* READ VARIABLE NAME OF LHS AND CHECK LEGALITY */
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF ALLOCATION(RECORD) THEN DO;
DO I=1 TO NITEM;
IF WORD=RECORD(1) THEN GO TO WORDOK;
END: IF I AND ERROR */
PUT FILE (SYSPRINT) EDIT (' *** NAME GIVEN HAS NOT BEEN ',
*DECLARED AS PART OF THE RECORD') (SKIP,A(29),A(30));
ERROR=1; GO TO BADEND;
END:

/* READ TO =, LHS MAY INCLUDE PARENTHESESISED EXPRESSION */
WORDOK:
WORDST=SYNTAX; WORDSTC=SYNCT; /* STORE NAME */
BRSTC=0;
READEQ:
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF SYNTAX(1)="=" THEN DO; /* READ PARENTHESESISED EXP */
SYNTAX(1)="="; BRSTC=1; BRRCNT+1; END:
READBRAC:
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF BRCNT+SYNCT>64 THEN DO;
PUT FILE (SYSPRINT) EDIT (' *** ARRAY SUBSCRIPT LENGTH ',
'EXCEEDS 64 CHARACTERS ALLOWED BY COMPILER')
(Skip,A(29),A(41));
ERROR=1; GO TO BADEND;
END:
DO I=1 TO SYNCT; /* SAVE NEW UNIT IN STRING */
BRST(BRRCNT+1)=SYNTAX(1);
END:
IF SYNTAX(1)="=" THEN DO;
BRCNT=BRRCNT+1; IF BRCNT=0 THEN GO TO READEQ; END:
IF SYNTAX(1)="=" THEN BRCCNT=BRRCNT+1;
GO TO READBRAC;
END:
IF SYNTAX(1)="=" THEN GO TO EQUALS; /* MUST BE ( OR = */
PUT FILE (SYSPRINT) EDIT (' *** ASSIGNMENT FORMAT INVALID')
(Skip,A(30));
ERROR=1; GO TO BADEND;
EQUALS: PUT FILE (OBJECT) EDIT (WORDST(1) DO I=1 TO WORDSTC)
(X(1),32 A(1));
/* PROCEDURE: FRED PRE-PROCESSOR, DAVID MENDUS, 18/3/67 */

DECLARE (ENDFLAG FIXED BINARY (1),
SYNTAX(32) PACKED CHARACTER (1),
SYNCNT FIXED BINARY,
ERROR FIXED BINARY ) STATIC EXTERNAL;

PUT FILE (OBJECT) EDIT (' DECLARE') (A(8));

GETNEXT: CALL ENTITY;
  IF ENDFLAG THEN DO:
    PUT FILE (OBJECT) EDIT (' :) (A(2)); RETURN; END;
  IF SYNTAX(1)="('" THEN DO:
    PUT FILE (OBJECT) EDIT (' (') (A(2));
    CALL BRACKET; GO TO GETNEXT; END;
  ELSE DO:
    PUT FILE (OBJECT) EDIT (' (SYNTAX(1) DO I=1 TO SYNCNT))
      (X(I),32 A(I)); GO TO GETNEXT; END;

END DCST;
PROCEDURE:  **FRED PRE-PROCESSOR, DAVID MENOUS, 25/2/67**  
ROUTINE TO PROCESS A CONTINUE ON A ** STATEMENT. NOTE THAT  
THESE STATEMENTS DO NOT GENERATE ANY OBJECT CODE.  **/

DECLARE ERROR FIXED BINARY(1);  
ENDFLAG  FIXED BINARY(1);  ) STATIC EXTERNAL:  

** NO OTHER NON-BLANK CHARACTERS SHOULD BE PRESENT.  **/  
CALL ENTITY:  
IF ENDFLAG=1 THEN RETURN;  
PUT FILE(SYSPRINT) EDIT (' *** EXCESS CHARACTERS IN THIS **/  
'CONTINUE STATEMENT HAVE BEEN DELETED') (SKIP,A(31),A(36));  
ERROR=1;  
DO WHILE (ENDFLAG=1):  
CALL ENTITY;  
END;  
RETURN:

END COST:
PUT FILE (OBJECT) EDIT ('' =1;') (A(4));
    IF LSTFLG=-1 THEN DO: *;
      PUT FILE (OBJECT) EDIT ('' PUT FILE ('', (FILENAME) DO I=1 TO
      FILECNT)) (A(11), 32, A(1)));
      PUT FILE (OBJECT) EDIT ('' EDIT ('' RECORD ADJUSTED'' ')',
        'SKIP(2),A(16))'')) (A(29), A(15));
    END:
    PUT FILE (OBJECT) EDIT ('' GO TO Q_#COMPLT; END;'') (A(22));
    /* DEAL WITH VALID RECORD */
    PUT FILE (OBJECT) EDIT ((FLGWORD(I) DO I=1 TO FLGCNT))
      (32, A(1));
    PUT FILE (OBJECT) EDIT (''-0; Q_#COMPLT:*') (A(14));
    /* WRITE CODE TO WRITE ERROR LISTINGS */
    IF LSTFLG=-1 THEN GO TO NOCODE:
      IF PRLFNG THEN PUT FILE (OBJECT) EDIT ('' DO:'') (A(4));
      ELSE PUT FILE (OBJECT) EDIT ('' IF L#0 FATAL)) (Q_#QUERY)''',
        'ADJUST) THEN DO: *= (A(25), A(27));
      IF PRLFNG THEN PUT FILE (OBJECT) EDIT ('' EDIT ) (') (A(8));
      ELSE PUT FILE (OBJECT) EDIT ('' LIST ) (') (A(8));
      PUT FILE (OBJECT) EDIT (''LISTSTR(I) DO I=1 TO LISTCNT)''',
        (201, A(1));
      IF PRLFNG THEN PUT FILE (OBJECT) EDIT (''R(Q_#FORM)''
        (A(12));
      PUT FILE (OBJECT) EDIT (''' END;''') (A(6));
      PUT FILE (OBJECT) EDIT ('' IF Q_DIGCT=0 THEN DO;'' (A(24));
      PUT FILE (OBJECT) EDIT ('' EDIT ) (') (FILENAME) DO I=1 TO
      FILECNT))''', (A(11), 32, A(1));
      PUT FILE (OBJECT) EDIT (''' EDIT (Q_#DIDAG(1Q_#) DO IQ_#=1',
        TO 12)) (SKIP, A(40))''')) (A(32), A(27));
      PUT FILE (OBJECT) EDIT (''' END;''') (A(5));
      /* CHECK NAME OF EDIT IS CORRECT */
      CALL ENTITY: IF ENDFLAG THEN GO TO NONAME;
      IF NOT EDITNAME THEN DO:
        PUT FILE (SYSPRNT) EDIT ('' *** NAME GIVEN CANNOT BE ',
        'RECOGNISED'') (SKIP, A(26), A(10));
      END;
      NONAME: PUT FILE (OBJECT) EDIT (''' END 'EDITNAME', ''')
        (A(5), A(7), A(1));
      /* IF CROSS REFERENCE TABLE WAS FILLED THEN WRITE IT */
      IF ALLOCATION(RECORD) THEN DO:
        PUT FILE (SYSPRNT) EDIT (''' LOGICAL CROSS-REFERENCE TABLE '',
        'FOR THIS EDIT IS') (SKIP(60), A(131), A(16));
        DO I=1 TO NITEM:
          PUT FILE (SYSPRNT) EDIT (RECORD(I), '' ****') (SKIP, A(12),
            A(5));
        END;
        /* COUNT OF NAMES ON LINE */
        DO K=1 TO NITEM:
          IF XREP(F,K)='' THEN DO:
            J=J+1; IF J=0 THEN DO:
              PUT FILE (SYSPRNT) EDIT (''' SKIP,X(16), A(1))'';
            J=1;
          END;
        END;
        PUT FILE (SYSPRNT) EDIT (RECORD(K)) (X(1), A(12));
      END;
      END;
      PUT FILE (SYSPRNT) SKIP(60);
COMPLT: PROCEDURE;
/* FRED PRE-PROCESSOR, DAVID MENDUS, 27/3/67
TO COMPLETE THE OBJECT CODE OF ONE EDIT */

DECLARE (ENDFLAG FIXED BINARY(1),
SYNTAX(32) PACKED CHARACTER(1),
SYNCT FIXED BINARY,
ERROR FIXED BINARY(1),
LISTFLG FIXED BINARY(1),
LISTCNT FIXED BINARY,
FORELG FIXED BINARY(1),
PRIFLAG FIXED BINARY(1),
MAXVAL FIXED BINARY,
FILENAME(32) PACKED CHARACTER(1),
FILECNT FIXED BINARY,
NITEM FIXED BINARY,
FLGodo(32) PACKED CHARACTER(1),
FLGcnt FIXED BINARY,
EDITNAM CHARACTER(7),
STATIC EXTERNAL,
RECORD(NITEM) PACKED CHARACTER(12) EXTERNAL CONTROLLED,
REFLAG(NITEM) BIT(1) CONTROLLED EXTERNAL,
LISTSTR(LISTCNT) PACKED CHARACTER(1) EXTERNAL CONTROLLED,
WORD CHARACTER(32) DEFINED SYNTAX;

/* WRITE CODE TO DEAL WITH FATAL ERROR RECORDS */
PUT FILE (OBJECT) EDIT ('# IF O #ADJUST>',LMAXVAL,' THEN ')
'Q #FATAL=1;') (A(15),F(1,0),A(6),A(11));
PUT FILE (OBJECT) EDIT ('# IF O #FATAL THEN DO: ',(FLGWORD(1)
DO I=1 TO FLGcnt) (A(22),32 A(11));
IDE FILE (OBJECT) EDIT ('#3:1') (A(4));
IF ALLOCATION (RECORD) THEN DO;
DO I=1 TO NITEM:
IDE FILE (OBJECT) EDIT (RECORD(1),'=O_#',RECORD(1),'#;')
(A(12),A(4),A(12),A(11));
END:
IF LISTFLG=*/1 THEN DO:
IDE FILE (OBJECT) EDIT ('# IF L #ADJUST>',LMAXVAL,' THEN ')
'Q #FATAL=1;') (A(15),F(1,0),A(6),A(11));
IDE FILE (OBJECT) EDIT ('# IF L #FATAL THEN DO: ',(FLGWORD(1)
DO I=1 TO FLGcnt) (A(22),32 A(11));
IDE FILE (OBJECT) EDIT ('#3:') (A(4));
END:
IDE FILE (OBJECT) EDIT ('GO TO Q #COORD: END;') (A(22));

/* DEAL WITH QUERY MESSAGE */
IDE FILE (OBJECT) EDIT ('# IF Q #QUERY THEN DO: ',(FLGWORD(1)
DO I=1 TO FLGcnt) (A(21),32 A(11));
IDE FILE (OBJECT) EDIT ('#2:1') (A(4));
IF LISTFLG=*/1 THEN DO:
IDE FILE (OBJECT) EDIT ('# IF L #QUERY THEN DO: ',(FLGWORD(1)
DO I=1 TO FLGcnt) (A(21),32 A(11));
IDE FILE (OBJECT) EDIT ('#3:') (A(4));
END:
IDE FILE (OBJECT) EDIT ('GO TO Q #COORD: END;') (A(22));

/* DEAL WITH ADJUST MESSAGE */
IDE FILE (OBJECT) EDIT ('# IF O #ADJUST THEN DO: ',(FLGWORD(1)
DO I=1 TO FLGcnt) (A(22),32 A(11));
ELSE DO:
   LASTIF=0; LASTCOL=MATRIXCT;
   RETURN; END;
/
# COLUMN NOT THE SAME AS LAST. MUST BE TWO MORE IF LAST WAS IF/
NOTSAME: IF MATRIXCT=LASTCOL THEN DO:
   IF LASTIF=0 THEN GO TO ISTHISIF:
   IF (MATRIXCT-LASTCOL)-1 THEN GO TO ILLEGAL:
   IF LASTIF=0 THEN GO TO ILLEGAL:
   GO TO ISTHISIF:
   END;
/
# COLUMN LESS THAN LAST. MUST BE IF ACTIVE IN PRESENT COLUMN #/
IF IFFLG(MATRIXCT)=0 THEN GO TO BADIFBLK:
IF IFFLG(MATRIXCT)=1 THEN GO TO ELSEPROC:
PUT FILE(OBJECT) EDIT (" "GO TO "; LABELS(MATRIXCT)",&1)
(A(7),A(6),A(2)); /* WRITE GO TO TO SKIP AROUND #/
LABELS(MATRIXCT)=1:
IF IFFLG=27)(IFLAG=28) THEN DO; /* THIS IS PART OF IF #/
   FLG=28: END:
   ELSE DO:
   IF FLAG=5 THEN DO:
      IFFLG(MATRIXCT)=1: END:
      ELSE GO TO NOELSE;
      END:
      LASTIF=1; LASTCOL=MATRIXCT: RETURN:
#
# NO ELSE STATEMENT TO IF-GROUP IN PRESENT COLUMN #/
NOELSE: PUT FILE(SYSPRINT) EDIT (" *** NO ELSE STATEMENT TO IF- ",
   " ** GROUP CLOSING BY THIS STATEMENT " ) (SKIP,A(29),A(30));
ERROR=1:
DO IF=MATRIXCT TO 10; /* CLEAR ALL IF-FLAGS TO TRY AGAIN SAVE #/
   IFFLG=0: /* EXTRA ERRORS CAUSED BY THIS ONE #/
   LABELS(I)=0:
   END:
   LASTCOL=MATRIXCT: RETURN:
#
# THIS STATEMENT IS FIRST AFTER LEGAL IF/ELSE GROUP #/
ELSEPROC:DO IF=MATRIXCT TO 10; /* ALL HIGHER IF-BLOCKS MUST HAVE ELSE #/
   IF IFFLG(I)=1 THEN GO TO NOELSE;
   IF LASTIF=1 THEN DO: /* LABEL NEEDED, WRITE IT #/
      PUT FILE(OBJECT) EDIT (LABELS(I)",&1)(X(1),A(6),A(2)):
      LABELS(I)=LABELS(I)+1: /* UPDATE LABEL STORE #/
      PUT STRING (LABELS(I)) EDIT (" **,?,LABLECT(I))
(A(1),F(1,0),A(1),F(3,0)):
      LABELS(I)=0: IFFLAG(I)=0:
      END:
      END:
      GO TO ISTHISIF:
      END COLCHK:
COLCHK:  PROCEDURE:  
/* FIXED PROC-PROCESSOR DAVID WENDUS 4/3/67  
PROCEDURE TO CHECK THE STARTING COLUMN OF EACH STATEMENT  
FOR VALIDITY AND CONTROL THE IF-BLOCK STRUCTURE */

DECLARE (COLONE FIXED BINARY,  
FLAG FIXED BINARY,  
MATRXCT FIXED BINARY,  
ERROR FIXED BINARY(1) ) STATIC EXTERNAL,
(LASTCOL FIXED BINARY,  
IFFLG(10) FIXED BINARY(1),  
LASTIF FIXED BINARY(1),  
LABLS(10) CHARACTER(6),  
LABLC(10) FIXED BINARY,  
CONT CHARACTER(1),  
LABSET(10) FIXED BINARY(1),  
FIRSTTIME FIXED BINARY(1) INITIAL(0) ) STATIC:

/* INITIALISE LABEL ARRAYS AND FLAGS FIRST TIME ONLY. */
IF FIRSTTIME THEN GO TO CHECKCOL:
  LASTCOL.LASTIF.LABSET(1),LABSET(2),LABSET(3),LABSET(4),  
  LABSET(5),LABSET(6),LABSET(7),LABSET(8),LABSET(9),  
  LABCOL(1),IFFLG(3),IFFLG(4),IFFLG(5),  
  IFFLG(6),IFFLG(7),IFFLG(8),IFFLG(9),IFFLG(10)=0,  
  LABLC(1),LABLC(2),LABLC(3),LABLC(4),LABLC(5),LABLC(6),  
  LABLC(7),LABLC(8),LABLC(9),LABLC(10)=1001;  
  FIRSTTIME=1;

/* CHECK FOR LEGAL COLUMN AND CONVERT TO "MATRIX FORM" */
CHECKCOL:MATRXCT=1; DO I=7 TO 25 BY 2;  
  IF COLONE=1 THEN GO TO LEGAL:
    MATRXCT=MATRXCT+1;
  END;
  ILLEGAL:  PUT FILE (SYSPRINT) EDIT ("*** STARTING COLUMN IS ILLEGAL")  
    (SAP,ACRT);  
    ERROR=1; LASTCOL=0; LASTIF=0; RETURN;

/* CHECK COLUMN WITH COLUMN OF LAST STATEMENT */
LEGAL:  IF MATRXCT=LASTCOL THEN GO TO NUTSAME;
  IF IFBLKCHK IF IFFLG(MATRXCT)=0 THEN DO:
    BADIFBLK:  PUT FILE (SYSPRINT) EDIT ("*** IF-BLOCK STRUCTURE IS ILLEGAL")  
      (SAP,ACRT);  
      ERROR=1; LASTCOL=0; LASTIF=0; RETURN;
  END;
  ISTRISIF:IF FLAG=1 THEN DO:  
    /* IS THIS FIRST IF OF GROUP? */  
    IF IFFLG(MATRXCT)=1; LASTIF=1; LASTCOL=MATRXCT;  
    RETURN; END;
PROCEDURE: RHCKFT /* FRM FR TEST R, AVID TNOUS, 9/3/67 TO READ AND WRITE A PARENTHEISED EXPRESSION. USES ENTITY*/

DECLARE (ENDFLAG FIXED BINARY(1),
SYNTAX(32) PACKED CHARACTER(1),
INTEND FIXED BINARY(1),
SYNCT FIXED BINARY,
ERROR FIXED BINARY(1) ) STATIC EXTERNAL,
PARACT FIXED BINARY STATIC;

PARACT=1; /* INITIALISE COUNT */

CALLER: CALL ENTITY:
IF ENDFLAG THEN DO; /* END OF STATEMENT TOO SOON */
PUT FILE (SYSPRINT) EDIT (' *** UNMATCHED PARENTHESSES')
(SKIP,AT(26)); INTEND=1;
ERROR=1; RETURN; END;
PUT FILE (OBJECT) EDIT ((SYNTAX(1) DO I=1 TO SYNCT))
(X(1),32 A(1));
IF SYNCT=1 THEN GO TO CALLER;
IF SYNTAX(1)=1 THEN DO; /* SEARCH FOR */ OR {
PARACT=PARACT-1;
IF PARACT=0 THEN RETURN;
GO TO CALLER;

END;
IF SYNTAX(1)=1 THEN PARACT=PARACT+1;
GO TO CALLER;
END BRACKET;
AIST:  PROCEDURE; /* FRED PRE-PROCESSOR, DAVID MENDUS, 18/3/67
TO EDIT AND WRITE AN ASSIGNMENT STATEMENT */

DECLARE (ENDFLAG BINARY FIXED (1),
SYNTAX(32) PACKED CHARACTER (1),
SYNCNT BINARY FIXED,
ERROR BINARY FIXED (1)) STATIC EXTERNAL;

PUT FILE (OBJECT) EDIT ((SYNTAX(1) DO I=1 TO SYNCNT))
(X(1), 32 A(1)); /* WRITE FIRST WORD OF LHS */

LHS:    CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
        IF SYNTAX(1)='*' THEN GO TO RHS; /* IF LHS IS NOT ONE WORD */
        IF SYNTAX(1)=='!' THEN DO; /* THEN IT MUST BE ARRAY ELEMENT */
           PUT FILE (OBJECT) EDIT '(' '(' A(2));
           CALL BRACKET; GO TO LHS; END;
        ELSE DO:
           PUT FILE (SYSPRINT) EDIT '(' '*** LHS OF ASSIGNMENT ILLEGAL')
           (X(1), 32 A(2)); CALL ENTITY; IF ENDFLAG=0 THEN GO TO BEND; RETURN;
        END:

BADEND: CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
        END:

RHS:    PUT FILE (OBJECT) EDIT '(' '=' (A(2));
        CALL ENTITY; IF ENDFLAG=0 THEN GO TO DORHS;

TOOSOON: PUT FILE (SYSPRINT) EDIT '(' '*** ASSIGNMENT STATEMENT',
          'INCOMPLETE') (SKIP, A(26), A(10));
        ERROR=1; RETURN;

DORHS:   PUT FILE (OBJECT) EDIT ((SYNTAX(1) DO I=1 TO SYNCNT))
          (X(1), 32 A(1));
        CALL ENTITY;
        IF SYNTAX(1)='*' THEN DO;
           PUT FILE (OBJECT) EDIT '(' '(' A(2));
           CALL BRACKET; GO TO DORHS; END;
        ELSE IF ENDFLAG=0 THEN GO TO DORHS;
           PUT FILE (OBJECT) EDIT '(' '*** A(1)) RETURN;

END AIST;
PROCEDURE; /* FRED PRE-PROCESSOR, DAVID WENDUS, 19/2/67 
BEGIN PROCESSING AN ADJUST STATEMENT. ADON DOES THE 
DIAGNOSTIC PROCESSING. */

DECLARE (ENDFLAG FIXED BINARY (1), 
SYNTAX(32) PACKED CHARACTER (1), 
SYNCTL FIXED BINARY, 
ERROR FIXED BINARY (1), 
ITEM FIXED BINARY ) STATIC EXTERNAL, 
RECORD (ITEM) PACKED CHARACTER (12) CONTROLLED EXTERNAL, 
WORD CHARACTER (32) DEFINED SYNTAX;

/* CHECK THAT THE VARIABLE TO BE ADJUSTED IS VALID */ 
CALL ENTITY; IF ENDFLAG THEN GO TO TOOSOON;
IF ALLOCATION (RECORD) THEN DO;
  DO I=1 TO NITEM;
  IF WORD=RECORD (I) THEN GO TO WRITE;
  ERROR=1; GO TO BADEND;
END;

WRITE:
  PUT FILE (OBJECT) EDIT (' ** ADJUST=BAD ** ADJUST='), 
  (SYNTAX (I)) DO I=1 TO SYNCNT) (A (23), 32 A (1));
WHERESEQ:
  IF SYNTAX (I)='' THEN DO; /* PARENTHESES LEGAL */
  PUT FILE (OBJECT) EDIT (' END '); 
  CALL Bracket; GO TO WHERESEQ;
END;

RHS:
  Put FILE (OBJECT) EDIT (' = ') (A (2)); /* FOUND */
  CALL ENTITY;
  IF ENDFLAG THEN DO; /* STATEMENT INCOMPLETE */
  PUT FILE (SYSPRINT) EDIT (' *** ADJUST STATEMENT IS ', 
  (INCOMPLETE) ) (SKIP, A (25), 32 A (10));
  ERROR=1; RETURN;
END;

TOOSOON:
  Put FILE (SYSPRINT) EDIT (' *** ADJUST STATEMENT IS ', 
  (INCOMPLETE) ) (SKIP, A (25), 32 A (10));
  ERROR=1; RETURN;
END;

BADEND:
  CALL ENTITY; IF ENDFLAG=0 THEN GO TO BADEND; RETURN;
END ADST:
## ATTRIBUTE TABLE

<table>
<thead>
<tr>
<th>DCL NO.</th>
<th>IDENTIFIER</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>ABS</td>
<td>GENERIC, BUILTIN FUNCTION</td>
</tr>
<tr>
<td>2</td>
<td>BILLED</td>
<td>AUTOMATIC, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>1</td>
<td>BILLER</td>
<td>ENTRY, DECIMAL, FLOAT(SINGLE)</td>
</tr>
<tr>
<td>2</td>
<td>CARD</td>
<td>AUTOMATIC, STRING, CHARACTER</td>
</tr>
<tr>
<td>2</td>
<td>COST</td>
<td>AUTOMATIC, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>2</td>
<td>CUSTOMER</td>
<td>AUTOMATIC, DECIMAL, FIXED(5,0)</td>
</tr>
<tr>
<td>2</td>
<td>DISCOUNT</td>
<td>AUTOMATIC, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>2</td>
<td>DISCRETE</td>
<td>AUTOMATIC, DECIMAL, FIXED(5,0)</td>
</tr>
<tr>
<td>10</td>
<td>ENDING</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>2</td>
<td>ERRCODE</td>
<td>AUTOMATIC, BINARY, FIXED(15,0)</td>
</tr>
<tr>
<td>2</td>
<td>FREIGHT</td>
<td>AUTOMATIC, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>2</td>
<td>FRTRATE</td>
<td>AUTOMATIC, DECIMAL, FIXED(5,2)</td>
</tr>
<tr>
<td>2</td>
<td>INSTRATE</td>
<td>AUTOMATIC, DECIMAL, FIXED(5,0)</td>
</tr>
<tr>
<td>2</td>
<td>INSURANCE</td>
<td>AUTOMATIC, DECIMAL, FIXED(7,2)</td>
</tr>
<tr>
<td>2</td>
<td>INVOICE</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>2</td>
<td>IQ_#</td>
<td>AUTOMATIC, BINARY, FIXED(15,0)</td>
</tr>
<tr>
<td>37</td>
<td>ITEM</td>
<td>AUTOMATIC, DECIMAL, FIXED(5,0)</td>
</tr>
<tr>
<td>45</td>
<td>Q1#001</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>71</td>
<td>Q1#002</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>71</td>
<td>Q1#003</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>79</td>
<td>Q1#004</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>95</td>
<td>Q1#005</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>111</td>
<td>Q1#006</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>123</td>
<td>Q1#007</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>129</td>
<td>Q1#008</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>71</td>
<td>Q2#001</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>DCL NO.</td>
<td>IDENTIFIER</td>
<td>ATTRIBUTES</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>95</td>
<td>Q#002</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>111</td>
<td>Q#003</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>12</td>
<td>Q#ADJUST</td>
<td>AUTOMATIC,BINARY,FIXED(15,0)</td>
</tr>
<tr>
<td>13</td>
<td>Q#BILLED</td>
<td>AUTOMATIC,DECIMAL,FIXED(7,2)</td>
</tr>
<tr>
<td>162</td>
<td>Q#COMPLT</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td>13</td>
<td>Q#COST</td>
<td>AUTOMATIC,DECIMAL,FIXED(7,2)</td>
</tr>
<tr>
<td>13</td>
<td>Q#CUSTOMER</td>
<td>AUTOMATIC,DECIMAL,FIXED(5,0)</td>
</tr>
<tr>
<td>12</td>
<td>Q#DIAG</td>
<td>(*) AUTOMATIC,PACKED,STRING,CHARACTER</td>
</tr>
<tr>
<td>12</td>
<td>Q#DIGIT</td>
<td>AUTOMATIC,BINARY,FIXED(15,0)</td>
</tr>
<tr>
<td>13</td>
<td>Q#DISCOUNT</td>
<td>AUTOMATIC,DECIMAL,FIXED(7,2)</td>
</tr>
<tr>
<td>13</td>
<td>Q#DISC RATE</td>
<td>AUTOMATIC,DECIMAL,FIXED(5,0)</td>
</tr>
<tr>
<td>12</td>
<td>Q#FATAL</td>
<td>AUTOMATIC,BINARY,FIXED(1,0)</td>
</tr>
<tr>
<td>13</td>
<td>Q#FREIGHT</td>
<td>AUTOMATIC,DECIMAL,FIXED(7,2)</td>
</tr>
<tr>
<td>13</td>
<td>Q#FRTRATE</td>
<td>AUTOMATIC,DECIMAL,FIXED(5,2)</td>
</tr>
<tr>
<td>13</td>
<td>Q#INS RATE</td>
<td>AUTOMATIC,DECIMAL,FIXED(5,0)</td>
</tr>
<tr>
<td>13</td>
<td>Q#INSURANCE</td>
<td>AUTOMATIC,DECIMAL,FIXED(7,2)</td>
</tr>
<tr>
<td>13</td>
<td>Q#ITEM</td>
<td>AUTOMATIC,DECIMAL,FIXED(5,0)</td>
</tr>
<tr>
<td>12</td>
<td>Q#QUERY</td>
<td>AUTOMATIC,BINARY,FIXED(1,0)</td>
</tr>
<tr>
<td>13</td>
<td>Q#UNITQTY</td>
<td>AUTOMATIC,DECIMAL,FIXED(5,0)</td>
</tr>
<tr>
<td>13</td>
<td>Q#UNITVAL</td>
<td>AUTOMATIC,DECIMAL,FIXED(7,2)</td>
</tr>
<tr>
<td>6</td>
<td>READ</td>
<td>STATEMENT LABEL CONSTANT</td>
</tr>
<tr>
<td></td>
<td>SYSIN</td>
<td>FILE,EXTERNAL</td>
</tr>
<tr>
<td></td>
<td>SYSPRINT</td>
<td>FILE,EXTERNAL</td>
</tr>
<tr>
<td>2</td>
<td>UNITQTY</td>
<td>AUTOMATIC,DECIMAL,FIXED(5,0)</td>
</tr>
<tr>
<td>2</td>
<td>UNITVAL</td>
<td>AUTOMATIC,DECIMAL,FIXED(7,2)</td>
</tr>
</tbody>
</table>
COMPILER DIAGNOSTIC MESSAGES

THE SUPPRESSION OF WARNING MESSAGES HAS BEEN REQUESTED.

TIME FOR THIS COMPILATION WAS 1.14 MINUTES
<table>
<thead>
<tr>
<th>Record in Error</th>
<th>Billed Cost Does Not Agree</th>
<th>Cost Invalid for Quantity</th>
<th>Record Adjusted</th>
<th>Discount Recalculated</th>
<th>Record in Error</th>
<th>Billed Cost Does Not Agree</th>
<th>Discount Rate Invalid</th>
<th>Item Code Out of Range</th>
<th>Record Queried</th>
<th>Freight High</th>
<th>Record in Error</th>
<th>Billed Cost Does Not Agree</th>
<th>Customer Code Out of Range</th>
<th>Record in Error</th>
<th>Billed Cost Does Not Agree</th>
<th>Discount Rate Invalid</th>
<th>Insurance Miscalculated</th>
<th>Freight High</th>
<th>Record in Error</th>
<th>Billed Cost Does Not Agree</th>
<th>Record Adjusted</th>
<th>Insurance Recalculated</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>1200.00</td>
<td>0</td>
<td>0.00</td>
<td>15.65</td>
<td>0.00</td>
<td>137</td>
<td>12</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>10</td>
<td>34.26</td>
<td>0.00</td>
<td>100.00</td>
<td>100</td>
<td>10</td>
<td>73.00</td>
<td>100.00</td>
<td>60.00</td>
<td>100.00</td>
<td>0.16</td>
<td>0.00</td>
</tr>
<tr>
<td>21.70</td>
<td>34.26</td>
<td>0.06</td>
<td>0.00</td>
<td>30.734</td>
<td>0.00</td>
<td>486</td>
<td>10</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>180.00</td>
<td>2.46</td>
<td>0.00</td>
<td>185.00</td>
<td>100</td>
<td>100</td>
<td>73.00</td>
<td>100.00</td>
<td>0.00</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>1.34</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>486.1</td>
<td>2</td>
<td>1.00</td>
<td>3.00</td>
<td>0.00</td>
<td>100.00</td>
<td>1.42</td>
<td>0.00</td>
<td>100.00</td>
<td>5</td>
<td>5</td>
<td>73.00</td>
<td>100.00</td>
<td>0.00</td>
<td>100.00</td>
<td>0.16</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>3.65</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>998</td>
<td>0.42</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>99997</td>
<td>0.02</td>
<td>0.00</td>
<td>100.00</td>
<td>0.00</td>
<td>100</td>
<td>139.61</td>
<td>100.00</td>
<td>0.00</td>
<td>100.00</td>
<td>0.2</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>376</td>
<td>1.35</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>27.00</td>
<td>0.00</td>
<td>0.00</td>
<td>30.31</td>
<td>0.00</td>
<td>5</td>
<td>30.31</td>
<td>100.00</td>
<td>0.00</td>
<td>100.00</td>
<td>0.07</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>0.96</td>
<td>0.07</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1031</td>
<td>1.39</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>13.90</td>
<td>1.17</td>
<td>1.17</td>
<td>514.60</td>
<td>0.00</td>
<td>2</td>
<td>2.45</td>
<td>100.00</td>
<td>0.00</td>
<td>100.00</td>
<td>2.11</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>3.51</td>
<td>0.02</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>998</td>
<td>0.06</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>16.32</td>
<td>1.35</td>
<td>1.35</td>
<td>32.10</td>
<td>0.00</td>
<td>5</td>
<td>32.10</td>
<td>100.00</td>
<td>0.00</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>501</td>
<td>0.27</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>24.60</td>
<td>1.39</td>
<td>1.39</td>
<td>80.27</td>
<td>0.00</td>
<td>10</td>
<td>80.27</td>
<td>100.00</td>
<td>0.00</td>
<td>100.00</td>
<td>2.66</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>2576</td>
<td>0.52</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>30.52</td>
<td>30.52</td>
<td>32.10</td>
<td>0.00</td>
<td>20</td>
<td>32.10</td>
<td>100.00</td>
<td>0.00</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>2581</td>
<td>2.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1605.40</td>
<td>80.27</td>
<td>80.27</td>
<td>32.10</td>
<td>0.00</td>
<td>20</td>
<td>32.10</td>
<td>100.00</td>
<td>0.00</td>
<td>100.00</td>
<td>2.66</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1637.50</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1637.50</td>
<td>0.00</td>
<td>2</td>
<td>1637.50</td>
<td>100.00</td>
<td>0.00</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
BILLY:  PROCEDURE OPTIONS(MAIN);  PAGE 3

01  Q1  THEN  DO:  GO TO, Q1007:  END:  IF COST / FREIGHT < 1.01 THEN  DO:  Q # QUER
12  Y:  IF Q # QDC# THEN Q # D16CT= 3016CT+1:  Q # D1AG# # D1ACT= "FREIGHT MIGHT":
13  IF ( COST = 0 ) THEN  DO:  FTRRATE= Q # FTRRATE= Q:  GO
15  TO Q1007:  END:  IF Q # # FATAL THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
17  THEN Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
19  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
21  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
23  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
25  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
27  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
29  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
31  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
33  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
35  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
37  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
39  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
41  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
43  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
45  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
47  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
49  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
51  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
53  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
55  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
57  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
59  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
61  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
63  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
65  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
67  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
69  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
71  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
73  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
75  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
77  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
79  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
81  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
83  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
85  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
87  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
89  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
91  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
93  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
95  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
97  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
99  TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
101 TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
103 TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
105 TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
107 TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
109 TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
111 TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
113 TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
115 TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
117 TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
119 TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
121 TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
123 TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
125 TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
127 TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.
129 TO Q # FATAL=1:  IF ( Q # FATAL ) THEN DO:  ERRCODE = 1:  ITEM = Q # ITEM:  PROT.