**7. SUBPROGRAMS AND LINKAGE**

**7.1 Introduction to Subprograms and Linkage**

**7.2 COBOL Calling an External Subprogram**

**7.3 COBOL Called as an External Subprogram**

**7.4 Assembler Calling an External Subprogram**

**7.5 Assembler Called as External Subprograms**

**7.6 COBOL EXEC Statement Parameters**

**7.7 Assembler Standard Linkage**

**7.8 Return Codes**

**7.9 Example Subprogram Linkage Job**

**7.1 Introduction to Subprograms**

When we refer to a subprogram, we are referring to a completely separate program and not an internal sub-routine or, in COBOL, a paragraph.

A program that calls, or passes control to, a subprogram may do so either statically or dynamically; there are advantages and disadvantages to both methods.

**Static Calls**

* The subprogram becomes a physical part of the calling program's load module at linkage time and it is larger because of this fact.
* If the subprogram requires changes in the future, the calling program AND the statically-called subprogram require re-linking.
* Future changes, or enhancements, to the subprogram will not be automatically implemented which could lead to processing problems.

**Dynamic Calls**

* The subprogram is a separate load module that has been individually linked and stored as a member in a load library.
* Changes to the subprogram do not require a re-linking with the calling program.
* Future changes, or enhancements, to the subprogram will be automatically implemented.

**7.2 COBOL Calling an External Subprogram**

COBOL offers simple ways to designate either a static subprogram call or a dynamic subprogram call.

**COBOL Static Call**

* Example of a COBOL static subprogram call not passing parameter(s):

CALL 'SUBPGM'.

* Example of a COBOL static subprogram call passing three parameters:

CALL 'SUBPGM' USING FIELD-1  
 FIELD-2  
 FIELD-3.

In this example, FIELD-1, FIELD-2 and FIELD-3 are fields of any data type and they can be declared anywhere within the DATA DIVISION.

**COBOL Dynamic Call**

* Example of a COBOL dynamic subprogram call not passing parameter(s):

01 SUBPROGRAM PIC X(8) VALUE 'SUBPGM'.

CALL SUBPROGRAM.

or

01 SUBPROGRAM PIC X(8).

MOVE 'SUBPGM' TO SUBPROGRAM.  
 CALL SUBPROGRAM.

SUBPROGRAM does NOT have to be a 01-level item.

* Example of a COBOL dynamic subprogram call passing two parameters:

01 SUBPROGRAM PIC X(8) VALUE 'SUBPGM'.  
  
 CALL SUBPROGRAM USING FIELD-A  
 FIELD-B.  
 or

01 SUBPROGRAM PIC X(8).

MOVE 'SUBPGM' TO SUBPROGRAM.  
 CALL SUBPROGRAM USING FIELD-A  
 FIELD-B.

Once again, FIELD-A and FIELD-B are fields of any data type and they can be declared anywhere within the DATA DIVISION.

SUBPROGRAM does NOT have to be a 01-level item.

**7.3 COBOL Called as an External Subprogram**

A COBOL subprogram that is passed parameters from a calling program must have a LINKAGE SECTION added to the DATA DIVISION. It is very common to place the LINKAGE SECTION right before the PROCEDURE DIVISION statement.

Example with three parameters being passed into the subprogram:

LINKAGE SECTION.

01 FIELD-1 PIC S9(4)V99 BINARY SYNC.  
 01 FIELD-2 PIC S9(3)V9(4) PACKED-DECIMAL.  
 01 FIELD-3 PIC X(10).

PROCEDURE DIVISION USING FIELD-1  
 FIELD-2  
 FIELD-3.

With the above, FIELD-1, FIELD-2 and FIELD-3 can be referenced anywhere within the subprogram's PROCEDURE DIVISION. And, if necessary, values can be moved to any one or more of the fields to be passed back to the caller.

A return code can be passed back to the caller by moving a numeric value to the COBOL special register named RETURN-CODE. For example:

MOVE 0 TO RETURN-CODE.

End ALL COBOL subprograms with GOBACK. instead of STOP RUN.

**7.4 Assembler Calling an External Subprogram**

Example of a static call and its preparation:  
  
 LA 1,PARMS R1 -> PARM LIST  
 L 15,=V(SUBPGM) R15 = ADDRESS OF SUBPGM   
 BALR 14,15 BRANCH TO SUBPGM  
  
Example of the parameter list referencing three fields:  
  
PARMS DC A(FIELD1) FULLWORD WITH ADDR OF FIELD1  
 DC A(FIELD2) FULLWORD WITH ADDR OF FIELD2  
 DC A(FIELD3) FULLWORD WITH ADDR OF FIELD3  
\*  
FIELD1 DC F'35' FULLWORD OF 35  
FIELD2 DC CL8'CSCI 465' 8-BYTE CHARACTER FIELD  
FIELD3 DC PL4'34.99' 4-BYTE PACKED FIELD OF 34.99

The set-up of the storage that is to be passed to a subprogram is prescribed by the conventions of standard linkage. Register 1 is loaded with the address of the first of a series of address constants, each a fullword of storage. The address constants hold the addresses of the actual fields being passed to the subprogram.

**Assembler Dynamic Call**

Note that there is a means by which an Assembler program can dynamically call a subprogram by using the Assembler LINK macro avalailable in SYS1.MACLIB.

**7.5 Assembler Called as an External Subprogram**

Once control is passed to an Assembler subprogram, the main object is to gain access to the parameters passed into it. After standard entry linkage, the subprogram will execute a load for a single parameter being passed or load multiple for more than one parameter being passed in.

Example of gaining access, or addressability, to the three parameters passed in from the above COBOL or Assembler calling program:

LM 2,4,0(1) R2 -> first parameter (FIELD1)  
\* R3 -> second parameter (FIELD2)  
\* R4 -> third parameter (FIELD3)

Remember that, upon entry to the subprogram, register 1 points to the first of a string of fullword address constants. The load multiple above will place the contents of the first address constant, i.e., the address of FIELD1, in register 2, the contents of the second address constant, i.e., the address of FIELD2, in register 3 and the contents of the third address constant, i.e., the address of FIELD3, in register 4.

It is important to remember that registers 2, 3 and 4 now point to the items where they are defined in the calling program and only their addresses have been passed to the subprogram.

To exit the subprogram, standard exit linkage should be used and, when necessary, a return code loaded into register 15. Be sure not to overwrite your return code in register 15 when you restore the caller's registers before branching on register 14 back to the caller.

**7.6 COBOL EXEC Statement Parameters**

Sometimes we need to pass a short parameter or two into a load module at the point of execution. Note that this is NOT considered a call to a subprogram and the program into which the EXEC-line parameters are passed can, in turn, pass them to a subprogram.

Example of passing two short EXEC-line parameters on a fetch step of JCL:

//JSTEP03 EXEC PGM=MAINPGM,PARM='CSCI 465/565 15'

Example of the COBOL program's Linkage Section to gain access to these EXEC-line parameters:

LINKAGE SECTION.

01 EXEC-LINE-PARM.  
 05 PARM-LENGTH PIC S9(4) COMP.  
 05 PARM-COURSE-INFO PIC X(15).  
 05 PARM-MAX-LINES PIC 9(2).

PROCEDURE DIVISION USING EXEC-LINE-PARM.

The PARM-LENGTH field above would automatically equal 17, the number of bytes in between the single quotes, or tick marks, of the EXEC-line parameter. This field can be checked in the receiving program to see if there are parameters being passed in or not.

As introduced in Chapter 1, parameters can be passed to any of the four common modules on the EXEC statement. For example, the BINDER is commonly passed the options PARM=MAP,LET,LIST. The EXEC statement allows passing parameters to ANY application programs as well. The operating system handles the passing of parameters from the EXEC statement as follows:

1. An area of storage is reserved in which to store the parameter values. The first data item in this storage area is a binary halfword containing the length of the entire parameter list passed in as one unit. The length field is calculated by summing up the total number of characters in the parameter list, including all special characters (like commas and quotes) except the outermost parentheses or quotes.
2. To follow standard linkage conventions, a parameter list containing the address of this storage area is built. In standard linkage, a parameter list contains only addresses, not actual data values.
3. R1 is set up as the parameter address list.

In Assembler, the parameter values could be accessed as follows:

\* R1 = ADDRESS OF STANDARD PARAMETER LIST

L 2,0(,1) R2 = ADDRESS OF THE STORAGE AREA

LH 3,0(,2) R3 = HALF WORD BINARY LENGTH FIELD

LA 4,2(,2) R4 = ADDRESS OF FIRST PARM VALUE

In COBOL, the EXEC parameters are accessed as if they were passed in by another application program. The EXEC parm data is described in the Linkage Section, and the Procedure Division header must contain the USING option. Remember to code the data field containing the total parameter length and, also, to code a description of all of the EXEC parm values, including special characters.

For an example, consider the following:

//JSTEP05 EXEC PGM=BOWLING,REGION=256K,COND=(0,NE),PARM=(200,080)

and in the program itself:

DATA DIVISION.

LINKAGE SECTION.

01 EXEC-PARMS.

05 PARM-LENGTH PIC S9(4) COMP SYNC.

05 NOMINAL-AVERAGE PIC 999.

05 FILLER PIC X.

05 HANDICAP-PERCENT PIC 9V99.

PROCEDURE DIVISION USING EXEC-PARMS.

The parameters passed in from the EXEC statement may be accessed via the data names in the Linkage Section. The application program should check the parameter list for valid data. If the length field in the above example is less than seven (a 3-digit average, a comma, and a 3-digit percent), then the parameters were not specified correctly. Also, since the data is defined as numeric, the program should check the fields for valid numeric data.

**7.7 Assembler Standard Linkage**

**Standard Entry Linkage**

The following is standard entry linkage to be used at the beginning of every Assembler program. Of course, the name MAIN would be different.

MAIN CSECT                                                          
         STM   14,12,12(13) SAVE CALLER'S REGS                     
         LR    12,15 SET R12 TO R15                         
         USING MAIN,12 ESTABLISH 1ST BASE REG                 
         LA    14,MAINSAVE R14 => CURRENT SAVE AREA               
         ST    13,4(,14) SAVE CALLER'S SAVE AREA ADDR           
         ST    14,8(,14) SAVE CURRENT SAVE AREA ADDR            
         LR    13,14 R13 => CURRENT SAVE AREA

**Explanation**

STM 14,12,12(13) saves all of the calling program’s registers, except for register 13, in the calling routine

LR 12,15 defines addressability in register 12 for the program

USING MAIN,12 sets register 12 as the base register for the program

LA 14,MAINSAVE points register 14 to an area in MAIN where its registers will be saved if this program, MAIN, calls another program

ST 14,8(13) saves the address of the current program's save area, MAINSAVE, into the caller's save area. This is sometimes referred to as the "forward pointer"

ST 13,4(14) saves the address of the caller program's save area into the current save area. This is sometimes referred to as the "backward pointer"

LR 13,14 points register 13 to the current program’s save area, MAINSAVE, readying it for a subprogram call where any subprogram would do exactly the same thing.

**Standard Exit Linkage**

L 13,4(,13) R13 => CALLER’S SAVE AREA

LM 14,12,12(13) RESTORE R14 THROUGH R12

BR 14 RETURN TO CALLER

If register 15 holds a return code that needs to be returned to the caller, use the following version of the standard exit linkage:

         L     13,4(,13) R13 => CALLER'S SAVE AREA  
         L     14,12(,13)     RESTORE R14  
         LM    0,12,20(13)     RESTORE R0 THROUGH R12  
         BR    14             RETURN TO CALLER

**7.8 Return Codes**

Coming soon

**7.9 Example Subprogram Linkage Job**

The example program on the following pages illustrates the passing of parameters via program linkage and the EXEC statement.

//KC0nnnnA JOB ,'LINKAGE EXAMPLE',MSGCLASS=H

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* \*

//\* THIS JOB COMPILES A COBOL MAIN PROGRAM, ASSEMBLES AN \*

//\* ASSEMBLER SUBPROGRAM, COMPILES A COBOL SUBPROGRAM, \*

//\* PRODUCES A SINGLE LOAD MODULE, AND EXECUTES THE LOAD \*

//\* MODULE. \*

//\* \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*

//JSTEP01 EXEC PGM=IGYCRCTL,PARM='FLAG(I,I),APOST'

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* \*

//\* 'STEP1' COMPILES THE COBOL MAIN PROGRAM \*

//\* \*

//\* STEPLIB LOCATION OF THE COBOL COMPILER (INPUT) \*

//\* SYSIN COBOL SOURCE PROGRAM (INPUT) \*

//\* SYSLIN OBJECT MODULE CREATED (OUTPUT) \*

//\* SYSPRINT MESSAGES FROM THE COMPILER (OUTPUT) \*

//\* \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*

//SYSIN DD \*

IDENTIFICATION DIVISION.

PROGRAM-ID. MAINPGM.

AUTHOR. TED PROGRAMMER.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* FUNCTION: THIS PROGRAM CONVERTS TEMPERATURES FROM \*

\* FAHRENHEIT TO CELSIUS OR VICE VERSA. \*

\* \*

\* INPUT: A FILE OF TEMPERATURES TO BE CONVERTED \*

\* OUTPUT: A REPORT OF THE CONVERTED TEMPERATURES IN \*

\* ALPHABETICAL ORDER BY CITY. \*

\* \*

\* ON ENTRY: PARAMETERS ARE PASSED IN ON THE EXEC \*

\* STATEMENT. SEE THE LINKAGE SECTION. \*

\* \*

\* ON EXIT: RETURN CODE IS ZERO. \*

\* \*

\* NOTES: NONE. \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ENVIRONMENT DIVISION.

INPUT-OUTPUT SECTION.

FILE-CONTROL.

SELECT TEMPERATURE-DATA ASSIGN TO TEMPFLE.

SELECT TEMPERATURE-REPORT ASSIGN TO RPTFLE.

DATA DIVISION.

FILE SECTION.

FD TEMPERATURE-DATA

RECORDING MODE IS F.

01 TEMPERATURE-RECORD.

05 CODE-IN PIC X.

05 CITY-IN PIC X(20).

05 HIGH-IN PIC S9(3)

SIGN IS LEADING SEPARATE.

05 LOW-IN PIC S9(3)

SIGN IS LEADING SEPARATE.

05 FILLER PIC X(51).

FD TEMPERATURE-REPORT

RECORDING MODE IS F.

01 REPORT-RECORD PIC X(133).

WORKING-STORAGE SECTION.

01 FLAGS.

05 EOF-FLAG PIC X VALUE 'N'.

88 END-OF-FILE VALUE 'Y'.

01 SUBSCRIPTS.

05 TEMP-SUB PIC S9(4) COMP SYNC

VALUE +1.

01 ACCUMULATORS.

05 NUM-OF-TEMPS PIC 99 VALUE 0.

05 LINE-CNTR PIC 99 VALUE 99.

01 WS-PARAMETERS.

05 WS-CODE PIC X VALUE SPACES.

05 WS-CITY PIC X(20) VALUE SPACES.

05 WS-HIGH PIC S9(3) COMP-3 VALUE +0.

05 WS-LOW PIC S9(3) COMP-3 VALUE +0.

05 WS-COUNT PIC 99 COMP VALUE 0.

01 WS-TEMP-TABLE.

05 CITY-DATA OCCURS 99.

10 CITY-NAME PIC X(20).

10 CITY-HIGH PIC S9(3) COMP-3.

10 CITY-LOW PIC S9(3) COMP-3.

01 HEADER1.

05 PIC X(53) VALUE SPACES.

05 PIC X(80) VALUE

'WORLDWIDE WEATHER SERVICE'.

01 HEADER2.

05 PIC X(60) VALUE SPACES.

05 DATE-OUT PIC X(10) VALUE SPACES.

05 PIC X(63) VALUE SPACES.

01 HEADER3.

05 PIC X(59) VALUE SPACES.

05 SCALE-OUT PIC X(20) VALUE SPACES.

05 PIC X(54) VALUE SPACES.

01 COLUMN-HEADER1.

05 PIC X(30) VALUE SPACES.

05 PIC X(4) VALUE 'CITY'.

05 PIC X(29) VALUE SPACES.

05 PIC X(4) VALUE 'HIGH'.

05 PIC X(27) VALUE SPACES.

05 PIC X(3) VALUE 'LOW'.

05 PIC X(34) VALUE SPACES.

01 HYPHENS.

05 PIC X(30) VALUE SPACES.

05 PIC X(20) VALUE ALL '-'.

05 PIC X(13) VALUE SPACES.

05 PIC X(4) VALUE ALL '-'.

05 PIC X(26) VALUE SPACES.

05 PIC X(4) VALUE ALL '-'.

05 PIC X(34) VALUE SPACES.

01 TEMPERATURE-LINE.

05 PIC X(30) VALUE SPACES.

05 CITY-OUT PIC X(20).

05 PIC X(10) VALUE SPACES.

05 HIGH-OUT PIC ++++++9.

05 PIC X(23) VALUE SPACES.

05 LOW-OUT PIC ++++++9.

05 PIC X(36) VALUE SPACES.

LINKAGE SECTION.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* 'LS-EXEC-PARMS' DESCRIBES THE DATA COMING INTO THE \*

\* PROGRAM VIA THE EXEC STATEMENT. \*

\* \*

\* 'LS-DATE' IS THE DATE FOR THE TEMPERATURE FORCAST \*

\* 'LS-CODE' DETERMINES THE SCALE USED FOR CONVERTING \*

\* THE INPUT TEMPERATURES: \*

\* F = FAHRENHEIT \*

\* C = CELSIUS \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

01 LS-EXEC-PARMS.

05 LS-LENGTH PIC S9(4) COMP SYNC.

05 LS-DATE PIC X(10).

05 FILLER PIC X.

05 LS-CODE PIC X.

PROCEDURE DIVISION USING LS-EXEC-PARMS.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* FUNCTION: CONTROLS THE MAIN FLOW OF LOGIC. \*

\* \*

\* PSEUDOCODE: \*

\* OPEN FILES. \*

\* READ FIRST RECORD. \*

\* INITIALIZE SUBSCRIPT. \*

\* DO WHILE (MORE INPUT RECORDS) \*

\* INCREMENT SUBSCRIPT \*

\* INVOKE '100-CONVERT-RTN' \*

\* END DO. \*

\* CALL 'SORTSUB' TO SORT THE TEMPERATURE \*

\* TABLE. \*

\* INVOKE '200-HEADER-RNT'. \*

\* INITIALIZE SUBSCRIPT. \*

\* DO WHILE (MORE ENTRIES IN TABLE) \*

\* INCREMENT SUBSCRIPT \*

\* INVOKE '300-PRINT-TABLE' \*

\* END-DO. \*

\* CLOSE FILES. \*

\* SET RETURN CODE TO ZERO \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

000-MAIN.

MOVE LS-DATE TO DATE-OUT.

IF LS-CODE = 'F'

MOVE ' FAHRENHEIT' TO SCALE-OUT

ELSE

MOVE ' CELSIUS' TO SCALE-OUT

END-IF.

OPEN INPUT TEMPERATURE-DATA

OUTPUT TEMPERATURE-REPORT.

READ TEMPERATURE-DATA

AT END MOVE 'Y' TO EOF-FLAG

END-READ.

PERFORM 100-CONVERT-RTN VARYING TEMP-SUB FROM 1 BY 1

UNTIL END-OF-FILE

OR TEMP-SUB > 99.

MOVE NUM-OF-TEMPS TO WS-COUNT.

CALL 'SORTSUB' USING WS-TEMP-TABLE

WS-COUNT.

PERFORM 300-PRINT-TABLE VARYING TEMP-SUB FROM 1 BY 1

UNTIL TEMP-SUB > NUM-OF-TEMPS.

CLOSE TEMPERATURE-DATA

TEMPERATURE-REPORT.

MOVE ZEROES TO RETURN-CODE.

GOBACK.

000-EXIT. EXIT.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* FUNCTION: CALLS AN EXTERNAL SUBPROGRAM TO CONVERT \*

\* THE HIGH AND LOW TEMPERATURES. \*

\* \*

\* PSEUDOCODE: \*

\* MOVE DATA TO PARM LIST. \*

\* CALL 'CONVSUB' TO CONVERT THE HIGH AND \*

\* LOW TEMPERATURES. \*

\* INCREMENT RECORD COUNTER. \*

\* READ THE NEXT INPUT RECORD. \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

100-CONVERT-RTN.

MOVE CODE-IN TO WS-CODE.

MOVE CITY-IN TO WS-CITY.

MOVE HIGH-IN TO WS-HIGH.

MOVE LOW-IN TO WS-LOW.

CALL 'CONVSUB' USING WS-PARAMETERS

LS-CODE

CITY-DATA (TEMP-SUB).

ADD 1 TO NUM-OF-TEMPS.

READ TEMPERATURE-DATA

AT END MOVE 'Y' TO EOF-FLAG

END-READ.

100-EXIT. EXIT.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* FUNCTION: PRINTS THE REPORT HEADERS \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

200-PRINT-HEADERS.

WRITE REPORT-RECORD FROM HEADER1 AFTER PAGE.

WRITE REPORT-RECORD FROM HEADER2 AFTER 1.

WRITE REPORT-RECORD FROM HEADER3 AFTER 1.

WRITE REPORT-RECORD FROM COLUMN-HEADER1 AFTER 2.

WRITE REPORT-RECORD FROM HYPHENS AFTER 1.

MOVE 0 TO LINE-CNTR.

200-EXIT. EXIT.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* FUNCTION: PRINTS THE REPORT DETAIL \*

\* \*

\* PSEUDOCODE: \*

\* IF PAGE IS FULL \*

\* INVOKE '250-PRINT-HEADINGS' \*

\* WRITE THE DETAIL LINE \*

\* INCREMENT THE LINE COUNTER. \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

300-PRINT-TABLE.

IF LINE-CNTR > 25

PERFORM 200-PRINT-HEADERS.

MOVE CITY-NAME (TEMP-SUB) TO CITY-OUT.

MOVE CITY-HIGH (TEMP-SUB) TO HIGH-OUT.

MOVE CITY-LOW (TEMP-SUB) TO LOW-OUT.

WRITE REPORT-RECORD FROM TEMPERATURE-LINE AFTER 2.

ADD 1 TO LINE-CNTR.

300-EXIT. EXIT.

/\*

//\*

//SYSLIN DD DSN=&&COBOBJ,SPACE=(CYL,(1,1)),DISP=(MOD,PASS)

//\*

//SYSPRINT DD SYSOUT=\*

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* \*

//\* 'SYSUT1' THRU 'SYSUT15' AND 'SYSMDECK' ARE WORK DATA \*

//\* SETS REQUIRED BY THE COBOL COMPILER V 5.1.0 \*

//\* \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*

//SYSUT1 DD SPACE=(CYL,(1,1))

//SYSUT2 DD SPACE=(CYL,(1,1))

//SYSUT3 DD SPACE=(CYL,(1,1))

//SYSUT4 DD SPACE=(CYL,(1,1))

//SYSUT5 DD SPACE=(CYL,(1,1))

//SYSUT6 DD SPACE=(CYL,(1,1))

//SYSUT7 DD SPACE=(CYL,(1,1))

//SYSUT8 DD SPACE=(CYL,(1,1))

//SYSUT9 DD SPACE=(CYL,(1,1))

//SYSUT10 DD SPACE=(CYL,(1,1))

//SYSUT11 DD SPACE=(CYL,(1,1))

//SYSUT12 DD SPACE=(CYL,(1,1))

//SYSUT13 DD SPACE=(CYL,(1,1))

//SYSUT14 DD SPACE=(CYL,(1,1))

//SYSUT15 DD SPACE=(CYL,(1,1))

//SYSMDECK DD SPACE=(CYL,(1,1))

//\*

//JSTEP02 EXEC PGM=ASMA90,PARM=ASA,COND=(0,LT)

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* \*

//\* 'STEP2' ASSEMBLES THE ASSEMBLER SUBPROGRAM \*

//\* \*

//\* STEPLIB LOCATION OF THE ASSEMBLER (INPUT) \*

//\* SYSLIB LOCATION OF ASSEMBLER MACROS (INPUT) \*

//\* SYSIN ASSEMBLER SOURCE PROGRAM (INPUT) \*

//\* SYSLIN OBJECT MODULE CREATED (OUTPUT) \*

//\* SYSPRINT MESSAGES FROM THE ASSEMBLER (OUTPUT) \*

//\* \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*

//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR

//\*

//SYSIN DD \*

PRINT NOGEN

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* FUNCTION: THIS SUBPROGRAM CONVERTS THE TEMPS FROM \*

\* FAHRENHEIT TO CELSIUS OR VICE VERSA \*

\* \*

\* ON ENTRY: R1 HAS ADDRESS OF PARAMETER LIST \*

\* 0(R1) = ADDR OF RECORD DATA: \*

\* CURRENT SCALE CODE LENGTH=1 \*

\* CITY NAME LENGTH=20 \*

\* HIGH TEMP LENGTH=2 S9(3) PACKED \*

\* LOW TEMP LENGTH=2 S9(3) PACKED \*

\* 4(R1) = ADDR OF CONVERSION CODE TO BE USED. \*

\* 8(R1) = ADDR OF TABLE ENTRY \*

\* \*

\* ON EXIT: THE CONVERTED TEMPERATURES AND THE \*

\* CORRESPONDING RECORD DATA IS STORED AT \*

\* THE TABLE EXTRY ADDRESS IN 8(R1) \*

\* \*

\* NOTES: NONE. \*

\* \*

\* PSEUDOCODE: \*

\* \*

\* STEP1: UNLOAD THE PARAMTERS \*

\* STEP2: CONVERT THE TEMPERATURES \*

\* STEP3: MOVE THE DATA INTO THE TABLE \*

\* STEP4: EXIT PROGRAM \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

CONVSUB CSECT SUBPROGRAM TO CONVERT TEMPS

STM 14,12,12(13) STANDARD LINKAGE

LR 12,15

USING CONVSUB,12

LA 14,SAVEAREA

ST 13,4(14)

ST 14,8(13)

LR 13,14

\*

\*\*\* STEP 1 \*\*\*

\*

LM 2,4,0(1) RECEIVE PARAMETERS

MVC CODE(1),0(2)

MVC CITY(20),1(2)

ZAP HIGH(2),21(2,2)

ZAP LOW(2),23(2,2)

MVC PARMCODE(1),0(3)

\*

\*\*\* STEP 2 \*\*\*

\*

ZAP HIGHWK(4),HIGH(2) PUT HIGH AND LOW IN

ZAP LOWWK(4),LOW(2) WORK FIELDS

CLC CODE(1),PARMCODE

BE NOTCODE

CLI CODE,C'C' CALCULATE CELSIUS?

BNE CALCFAHR IF NOT, CALCULATE FAHR

\*

MP HIGHWK(4),=P'18' HIGH TEMP

SRP HIGHWK(4),(64-1),5

AP HIGHWK(4),=P'32'

MP LOWWK(4),=P'18' CALCULATE CELSIUS

SRP LOWWK(4),(64-1),5 LOW TEMP

AP LOWWK(4),=P'32'

B NOTCODE

\*

CALCFAHR SP HIGHWK(4),=P'32' CALCULATE FAHRENHEIT

MP HIGHWK(4),=P'5' HIGH TEMP

DP HIGHWK(4),=P'9'

ZAP HOLDPK(3),HIGHWK(3)

ZAP HIGHWK(4),HOLDPK(3)

SP LOWWK(4),=P'32' CALCULATE FAHRENHEIT

MP LOWWK(4),=P'5' LOW TEMP

DP LOWWK(4),=P'9'

ZAP HOLDPK(3),LOWWK(3)

ZAP LOWWK(4),HOLDPK(3)

\*

\*\*\* STEP 3 \*\*\*

\*

NOTCODE ZAP HIGH(2),HIGHWK+2(2) MOVE HIGH TO WORK FIELD

ZAP LOW(2),LOWWK+2(2) MOVE LOW TO WORK FIELD

MVC 0(24,4),TABDATA MOVE CITY, HIGH & LOW

\* TO TABLE

\*\*\* STEP 4 \*\*\*

\*

L 13,4(13) EXIT LINKAGE

LM 14,12,12(13)

BR 14

\*

LTORG

ORG CONVSUB+((\*-CONVSUB+31)/32)\*32

DC C'\*\*\* STORAGE AREA FOR CONVSUB \*\*\*'

\*

SAVEAREA DC 18F'-1'

\*

TABDATA DS 0CL24 TABLE ENTRY

CITY DS CL20 CITY

HIGH DS PL2 HIGH

LOW DS PL2 LOW

CODE DS CL1 TEMP TYPE CODE

PARMCODE DS CL1 TEMP CONVERSION CODE

HIGHWK DC PL4'0' HIGH WORK FIELD

LOWWK DC PL4'0' LOW WORK FIELD

HOLDPK DC PL3'0' WORK FIELD

\*

END CONVSUB

/\*

//\*

//SYSLIN DD DSN=&&ASMOBJ,SPACE=(3040,(40,40),,,ROUND),DISP=(MOD,PASS)

//\*

//SYSPRINT DD SYSOUT=\*

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* \*

//\* 'SYSUT1' IS A WORK SET REQUIRED BY THE ASSEMBLER \*

//\* \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*

//SYSUT1 DD SPACE=(16384,(120,120),,,ROUND)

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* \*

//\* 'STEP3' COMPILES THE COBOL SUB PROGRAM \*

//\* \*

//\* STEPLIB LOCATION OF THE COBOL COMPILER (INPUT) \*

//\* SYSIN COBOL SOURCE PROGRAM (INPUT) \*

//\* SYSLIN OBJECT MODULE CREATED (OUTPUT) \*

//\* SYSPRINT MESSAGES FROM THE COMPILER (OUTPUT) \*

//\* \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*

//JSTEP03 EXEC PGM=IGYCRCTL,PARM='FLAG(I,I),APOST',COND=(0,LT)

//\*

//SYSIN DD \*

IDENTIFICATION DIVISION.

PROGRAM-ID. SORTSUB.

AUTHOR. TED PROGRAMMER.

DATE-WRITTEN. 02/16/2017.

DATE-COMPILED.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* FUNCTION: THIS SUBPROGRAM SORTS THE TEMPERATURE \*

\* TABLE IN ASCENDING ORDER BY CITY. \*

\* \*

\* INPUT: NONE. \*

\* OUTPUT: NONE. \*

\* \*

\* ON ENTRY: THE ADDRESS OF THE TEMPERATURE TABLE \*

\* AND THE ADDRESS OF THE NUMBER OF ENTRIES \*

\* IS PASSED INTO THE SUBPROGRAM. \*

\* SEE THE LINKAGE SECTION. \*

\* ON EXIT: THE TABLE CONTENTS WILL BE SORTED. \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ENVIRONMENT DIVISION.

DATA DIVISION.

WORKING-STORAGE SECTION.

01 TABLE-FIELDS.

05 SUB1 PIC S9(4) COMP SYNC VALUE +1.

05 SUB2 PIC S9(4) COMP SYNC VALUE +1.

05 COUNTER-1 PIC 99 VALUE 0.

05 COUNTER-2 PIC 99 VALUE 0.

01 TABLE-ENTRY-SAVE.

05 CITY-SAVE PIC X(20).

05 HIGH-SAVE PIC S9(3) COMP-3.

05 LOW-SAVE PIC S9(3) COMP-3.

LINKAGE SECTION.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* 'LS-TEMP-TABLE' IS THE TABLE IN THE MAIN PROGRAM. \*

\* 'LS-COUNT' IS THE NUMBER OF ENTRIES IN THE TABLE. \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

01 LS-TEMP-TABLE.

05 TABLE-ENTRY OCCURS 99.

10 CITY-NAME PIC X(20).

10 CITY-HIGH PIC S9(3) COMP-3.

10 CITY-LOW PIC S9(3) COMP-3.

01 LS-COUNT PIC 99 COMP.

PROCEDURE DIVISION USING LS-TEMP-TABLE

LS-COUNT.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* FUNCTION: CONTROLS THE MAIN FLOW OF LOGIC \*

\* \*

\* PSEUDOCODE: \*

\* INITIALIZE SUB1 TO BEGINNING OF TABLE. \*

\* DO WHILE (MORE THAN 1 TABLE ENTRY) \*

\* INVOKE '100-SORT-RTN' \*

\* INCREMENT SUB1 \*

\* END DO. \*

\* EXIT PROGRAM. \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

000-MAIN.

COMPUTE COUNTER-1 = LS-COUNT - 1.

PERFORM 100-SORT-RTN VARYING SUB1 FROM 1 BY 1

UNTIL SUB1 > COUNTER-1.

GOBACK.

000-EXIT. EXIT.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* FUNCTION: SORTS TABLE ENTRIES \*

\* \*

\* PSEUDOCODE: \*

\* INITIALIZE SUB2 TO ONE MORE THAN SUB1 \*

\* DO WHILE (MORE TABLE ENTRIES) \*

\* IF ENTRY1 IS GREATER THAN ENTRY2 \*

\* EXCHANGE DATA. \*

\* INCREMENT SUB2 \*

\* END DO. \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

100-SORT-RTN.

COMPUTE COUNTER-2 = SUB1 + 1.

PERFORM VARYING SUB2 FROM COUNTER-2 BY 1

UNTIL SUB2 > LS-COUNT

IF CITY-NAME(SUB2) < CITY-NAME(SUB1)

MOVE TABLE-ENTRY(SUB2) TO TABLE-ENTRY-SAVE

MOVE TABLE-ENTRY(SUB1) TO TABLE-ENTRY(SUB2)

MOVE TABLE-ENTRY-SAVE TO TABLE-ENTRY(SUB1)

END-IF

END-PERFORM.

100-EXIT. EXIT.

/\*

//\*

//SYSLIN DD DSN=&&SUBOBJ,SPACE=(CYL,(1,1)),DISP=(MOD,PASS)

//\*

//SYSPRINT DD SYSOUT=\*

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* \*

//\* 'SYSUT1' THRU 'SYSUT15' AND 'SYSMDECK' ARE WORK DATA \*

//\* SETS REQUIRED BY THE COBOL COMPILER V 5.1.0 \*

//\* \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*

//SYSUT1 DD SPACE=(CYL,(1,1))

//SYSUT2 DD SPACE=(CYL,(1,1))

//SYSUT3 DD SPACE=(CYL,(1,1))

//SYSUT4 DD SPACE=(CYL,(1,1))

//SYSUT5 DD SPACE=(CYL,(1,1))

//SYSUT6 DD SPACE=(CYL,(1,1))

//SYSUT7 DD SPACE=(CYL,(1,1))

//SYSUT8 DD SPACE=(CYL,(1,1))

//SYSUT9 DD SPACE=(CYL,(1,1))

//SYSUT10 DD SPACE=(CYL,(1,1))

//SYSUT11 DD SPACE=(CYL,(1,1))

//SYSUT12 DD SPACE=(CYL,(1,1))

//SYSUT13 DD SPACE=(CYL,(1,1))

//SYSUT14 DD SPACE=(CYL,(1,1))

//SYSUT15 DD SPACE=(CYL,(1,1))

//SYSMDECK DD SPACE=(CYL,(1,1))

//\*

//JSTEP04 EXEC PGM=HEWL,PARM='MAP,LET,LIST',COND=(0,LT)

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* \*

//\* 'STEP4' BINDS THE THREE OBJECT MODULES INTO ONE \*

//\* PROGRAM OBJECT. \*

//\* \*

//\* SYSLIN OBJECT MODULE TO BE BINDED (INPUT) \*

//\* SYSLIB PROGRAM OBJECT/OBJECT MODULE LIBRARY (INPUT) \*

//\* SYSLMOD THE CREATED PROGRAM OBJECT (OUTPUT) \*

//\* SYSPRINT MESSAGES FROM THE BINDER (OUTPUT) \*

//\* \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*

//SYSLIB DD DSN=CEE.SCEELKED,DISP=SHR

//\*

//SYSLIN DD DSN=&&COBOBJ,DISP=(OLD,DELETE)

// DD DSN=&&SUBOBJ,DISP=(OLD,DELETE)

// DD DSN=&&ASMOBJ,DISP=(OLD,DELETE)

//\*

//SYSLMOD DD DSN=KC0nnnn.CSCI465.LOADLIB(TEMPCONV),DISP=MOD

//\*

//SYSPRINT DD SYSOUT=\*

//\*

//SYSUT1 DD SPACE=(1024,(120,120),,,ROUND)

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* \*

//\* 'STEP5' EXECUTES THE LOAD MODULE CREATED IN 'STEP4' \*

//\* \*

//\* STEPLIB LOCATION OF LOAD MODULE TO BE EXECUTED \*

//\* INPUT THE TEMPERATURE INPUT FILE (INPUT) \*

//\* SYSUDUMP DEBUGGING AID (OUTPUT) \*

//\* CEEDUMP DEBUGGING AID (OUTPUT) \*

//\* REPORT OUTPUT FROM PROGRAM (OUTPUT) \*

//\* SYSOUT OUTPUT DEVICE (OUTPUT) \*

//\* SYSPRINT MESSAGES FROM SYSTEM (OUTPUT) \*

//\* \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*

//JSTEP05 EXEC PGM=TEMPCONV,COND=(0,LT),

// PARM='11-27-2016,C/TERMTHDACT(DUMP)'

//\*

//STEPLIB DD DSN=KC0nnnn.CSCI465.LOADLIB,DISP=SHR

//\*

//TEMPFLE DD DSN=KC0nnnn.CSCI465.CELSIUS,DISP=SHR

//\*

//RPTFLE DD SYSOUT=\*

//\*

//SYSUDUMP DD SYSOUT=\*

//\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\* \*

//\* 'STEP6' EXECUTES THE LOAD MODULE CREATED IN 'STEP4' \*

//\* \*

//\* STEPLIB LOCATION OF LOAD MODULE TO BE EXECUTED \*

//\* TEMPS THE TEMPERATURE DATA FILE (INPUT) \*

//\* SYSUDUMP DEBUGGING AID (OUTPUT) \*

//\* CEEDUMP DEBUGGING AID (OUTPUT) \*

//\* REPORT OUTPUT FROM PROGRAM (OUTPUT) \*

//\* SYSOUT OUTPUT DEVICE (OUTPUT) \*

//\* SYSPRINT MESSAGES FROM SYSTEM (OUTPUT) \*

//\* \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*

//JSTEP06 EXEC PGM=TEMPCONV,COND=(0,LT),

// PARM='11-27-2016,F/TERMTHDACT(DUMP)'

//\*

//STEPLIB DD DSN=KC0nnnn.CSCI465.LOADLIB,DISP=SHR

//\*

//TEMPFLE DD DSN=KC0nnnn.CSCI465.FAHREN,DISP=SHR

//\*

//RPTFLE DD SYSOUT=\*

//\*

//SYSUDUMP DD SYSOUT=\*

//\*

//

**The following could be used as the input data sets:**

Both should be set up as sequential data sets (or PDS members) with an LRECL=80.

KC0nnnn.CSCI465.CELSIUS (from the example above)

CWARSAW +007-007

CAMSTERDAM +002+000

CSAN DIEGO +013+010

CRAPID CITY -003-018

CPHOENIX +010+006

COMAHA +008+000

CMOSCOW -001-008

CHOUSTON +012+008

CHAVANA +040+037

CGUADALAJARA +032+010

CDALLAS +023+017

KC0nnnn.CSCI465.FAHREN (from the example above)

FWARSAW +046+018

FAMSTERDAM +036+032

FSAN DIEGO +055+050

FRAPID CITY +025+000

FPHOENIX +050+043

FOMAHA +048+032

FMOSCOW +030+016

FHOUSTON +054+048

FHAVANA +104+099

FGUADALAJARA +090+050

FDALLAS +075+064

**The following is produced by the above job:**

WORLDWIDE WEATHER SERVICE

11-27-2016

CELSIUS

CITY HIGH LOW

-------------------- ---- ----

AMSTERDAM +2 +0

DALLAS +23 +17

GUADALAJARA +32 +10

HAVANA +40 +37

HOUSTON +12 +8

MOSCOW -1 -8

OMAHA +8 +0

PHOENIX +10 +6

RAPID CITY -3 -18

SAN DIEGO +13 +10

WARSAW +7 -7

WORLDWIDE WEATHER SERVICE

11-27-2016

FAHRENHEIT

CITY HIGH LOW

-------------------- ---- ----

AMSTERDAM +36 +32

DALLAS +75 +64

GUADALAJARA +90 +50

HAVANA +104 +99

HOUSTON +54 +48

MOSCOW +30 +16

OMAHA +48 +32

PHOENIX +50 +43

RAPID CITY +25 +0

SAN DIEGO +55 +50

WARSAW +46 +18