Question?:

Workshop 7.2 - Enhancements to FAVRPT - **Report Statistics**

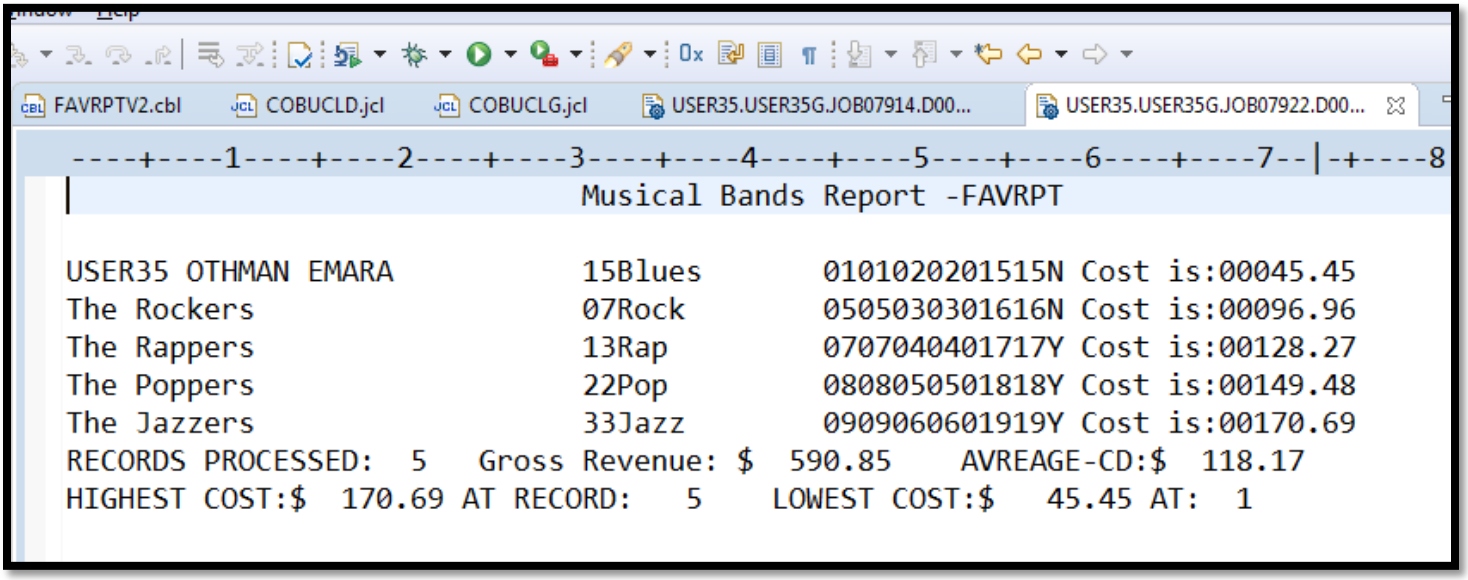
```
WORKING-STORAGE SECTION.
77  REC-KTR                    PIC 99 VALUE ZERO.
77  COST-TOTAL                 PIC 9(5)V99 VALUE ZERO.
77  REC-KTR-OUT                PIC Z9.
77  COST-TOTAL-OUT             PIC $999.99.
```

Modify FAVRPT as follows:

1. Add a WORKING-STORAGE SECTION with fields like the above - to support a few simple report statistics:
   1. A count of the number of records written; 2. A grand total of all CD sales; 3. The average CD sale.
   - *Utilize Numeric Edited Variable declarations. Your choice:* ***$, Zero Suppression, Commas, etc.***
2. Add the following code to the PROCEDURE DIVISION, after each FAVRPT-REC WRITE statement
   1. Add +1 to REC-KTR-OUT
   2. Add the calculated COST-CD to COST-TOTAL
3. At the end of the program (after all the input records have been read) display the following statistics:
   1. Number of records
   2. Gross Revenue (COST-TOTAL-OUT)
   3. Average CD Sale COST-TOTAL / Number of records

Optional/Challenge Exercise: Add code to find and calculate the Highest & Lowest CD cost. Print them out (DISPLAY them) at the end of the program with an appropriate message.

Answer : PROGRAME NAME FAVRPTV2




```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. FAVRPTV2.
      ***** This MODULE Workshop 5.3.1b - Create new COBOL program
      *****  THAT USES A FILE READ
      ***** VERSION 2 Workshop 7.2 THAT MAKE CLAUCLATION  PRINT
AVERAGE
       ENVIRONMENT DIVISION.
```

```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT FAVIN1  ASSIGN TO FAVIN
                   FILE STATUS IS FAVIN-F-STATUS.
    SELECT OUT2    ASSIGN TO FAVRPT.
DATA DIVISION.
FILE SECTION.
FD  FAVIN1
    RECORDING MODE IS F
    LABEL RECORDS ARE STANDARD
    RECORD CONTAINS 80 CHARACTERS
    BLOCK CONTAINS 0 RECORDS
    DATA RECORD IS FAVIN-REC.
01 FAVIN-REC.
   05 ARTIST-NAME                 PIC X(30).
   05 NUMBER-OF-MUSICIAN          PIC 9(02).
   05 MUSICAL-GENRE               PIC X(12).
   05 COST.
      10 CD-COST                  PIC 9(3)V99.
      10 SHIPPING-COST            PIC 9(2)V99.
      10 TAX                      PIC 9(2)V99.
   05 BAND-IS-STILL-TOGETHER      PIC X(1).
FD  OUT2
    RECORDING MODE IS F
    LABEL RECORDS ARE STANDARD
    RECORD CONTAINS 80 CHARACTERS
    BLOCK CONTAINS 0 RECORDS
    DATA RECORD IS FAVOUT-REC.
01 FAVOUT-REC.
   05 ARTIST-NAME-OUT             PIC X(30).
   05 NUMBER-OF-MUSICIAN-OUT      PIC 9(02).
   05 MUSICAL-GENRE-OUT           PIC X(12).
   05 COST-OUT.
      10 CD-COST-OUT              PIC 9(3)V99.
      10 SHIPPING-COST-OUT        PIC 9(2)V99.
      10 TAX-OUT                  PIC 9(2)V99.
   05 BAND-IS-STILL-TOGETHER-OUT  PIC X(1).
   05 COST-IS                     PIC X(9)    VALUE ' Cost is:'.
   05 COMPUTED-COST-OUT           PIC 9(5).99.

WORKING-STORAGE SECTION.
01 HEADER-1.
   05 FILLER                      PIC X(30)   VALUE SPACES.
```

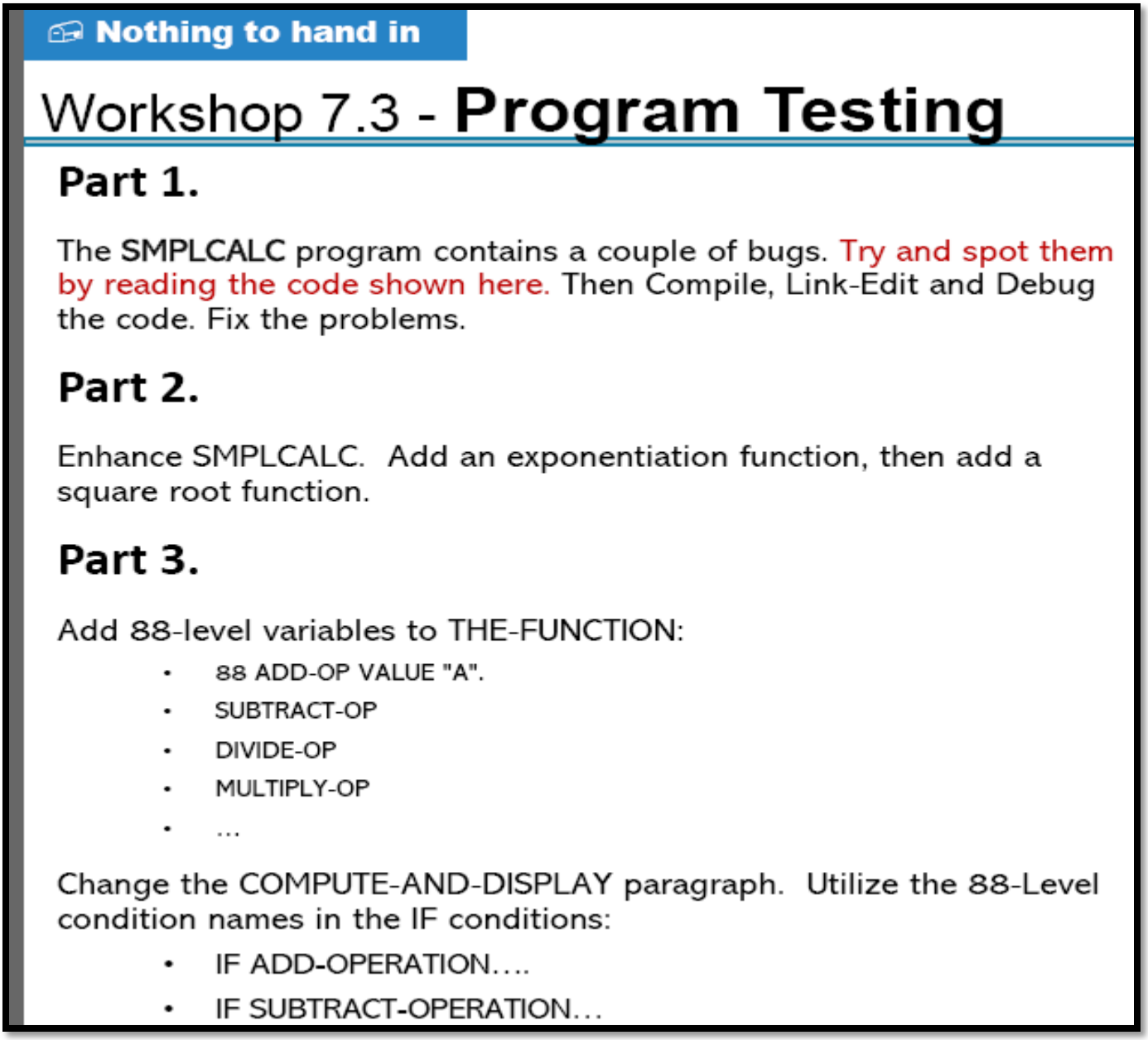
```
    05 FILLER PIC X(30) VALUE 'Musical Bands Report -FAVRPT'.
    05 FILLER                    PIC X(20)   VALUE SPACES.
 01 TRAILLER-1.
    05 FILLER PIC X(18)   VALUE 'RECORDS PROCESSED:'.
    05 REC-KTR-OUT         PIC ZZ9   .
    05 FILLER PIC X(18) VALUE '   Gross Revenue:'.
    05 COST-TOTAL-OUT      PIC $ZZZZZ.99   .
    05 FILLER  PIC X(15)   VALUE '    AVREAGE-CD:'.
    05 AVREAGE-CD-SALE-OUT   PIC $ZZZZZ.99   .
 01 TRAILLER-2.
    05 FILLER PIC X(13)       VALUE 'HIGHEST COST:'.
    05 CD-COST-HIGHEST-D      PIC $ZZZZZ.99    .
    05 FILLER PIC X(13)       VALUE ' AT RECORD: '.
    05 REC-NO-HIGHEST-D       PIC Z9  .
    05 FILLER PIC X(16)       VALUE '    LOWEST COST:'.
    05 CD-COST-LOWEST-D       PIC $ZZZZZ.99  .
    05 FILLER PIC X(5)       VALUE ' AT: '.
    05 REC-NO-LOWST-D         PIC Z9  .
*     JUST TO DEBUG ANY DATA OUT
 01 DEBUG-REC.
    05 FILLER    PIC X(5)    VALUE 'DEBUG'.
    05 FILLER    PIC X(13)   VALUE  ' file status:'.
    05 FAVIN-F-STATUS             PIC X(2).
    05 FILLER                     PIC X(4)    VALUE ' LR:'.
    05 LASTREC                    PIC X       VALUE SPACES.
       88 STELL-THERE-REC                     VALUE ' '.
       88 NO-MORE-RECORDS                     VALUE 'Y'.
    05   DBG-MESSAGE-ALL.
        10 DBG-MESSAGE.
            15 DBG-MSG1                 PIC X(15).
            15 DBG-MSG2                 PIC X(15).
        10 DBG-MSG3                  PIC X(10).
        10 DBG-MSG4                  PIC X(10).
        10 DBG-MSG5                  PIC X(5).
 01 COST-DEBUG.
        10 CD-COST-D                 PIC 9(3).99.
        10 FILLER                    PIC X(3)    VALUE ' + '.
        10 SHIPPING-COST-D           PIC 9(2).99.
        10 FILLER                    PIC X(3)    VALUE ' + '.
        10 TAX-D                     PIC 9(2).99.
*       COMPUTED DATAITEMS
 77 COMPUTED-COST                    PIC 9(5)v99.
 77 REC-KTR            PIC 99 VALUE ZEROS .
```

```
 77 REC-NO-HIGHEST      PIC 99 VALUE ZEROS .
 77 REC-NO-LOWEST       PIC 99 VALUE ZEROS .
 77 COST-TOTAL          PIC 9(5)V99 VALUE ZEROS .
 77 AVREAGE-CD-SALE     PIC 9(5)V99 VALUE ZEROS .
 77 CD-COST-HIGHEST     PIC 9(3)V99 VALUE ZEROS .
 77 CD-COST-LOWEST      PIC 9(3)V99 VALUE ZEROS .
 PROCEDURE DIVISION.
     OPEN INPUT FAVIN1.
     OPEN OUTPUT OUT2.
     WRITE FAVOUT-REC FROM HEADER-1.
     MOVE SPACES TO FAVOUT-REC.
     WRITE FAVOUT-REC AFTER ADVANCING 1 LINES.
*    Prime Read
     PERFORM READ-RECORD.
     PERFORM UNTIL LASTREC = 'Y' OR NO-MORE-RECORDS
             PERFORM PROCESS-RECORDS
             PERFORM WRITE-RECORD
             PERFORM READ-RECORD
     END-PERFORM
     PERFORM WRITE-LAST-REC
     PERFORM CLOSE-FILES
     STOP RUN.
 READ-RECORD.
     READ FAVIN1
*         AT END   MOVE 'Y' TO LASTREC
     AT END
        PERFORM END-OF-FILE
*    NOT AT END         PERFORM PROCESS-RECORDS
     END-READ.
 PROCESS-RECORDS.
     COMPUTE COMPUTED-COST =(CD-COST + SHIPPING-COST + TAX).
     COMPUTE REC-KTR = REC-KTR + 1 .
     IF  COMPUTED-COST > 0 AND REC-KTR = 1 THEN
         COMPUTE  CD-COST-HIGHEST = COMPUTED-COST
         COMPUTE  CD-COST-LOWEST = COMPUTED-COST
         COMPUTE  REC-NO-LOWEST = 1
         COMPUTE REC-NO-HIGHEST = 1
     END-IF.
*    Accumulate COST-TOTAL
     COMPUTE COST-TOTAL = (COST-TOTAL + COMPUTED-COST).
     MOVE CD-COST TO CD-COST-D .
     MOVE SHIPPING-COST TO SHIPPING-COST-D .
     MOVE TAX TO TAX-D .
```

```
     MOVE SPACES TO DBG-MESSAGE-ALL.
     PERFORM GET-HIGHEST-LOWEST-CD-COST.
*    MOVE ' P-RCD ' TO DBG-MSG1 .
*    WRITE FAVOUT-REC FROM DEBUG-REC.
*    Just to print debug data
*    MOVE COST-DEBUG TO DBG-MESSAGE .
*    WRITE FAVOUT-REC FROM DEBUG-REC.
 WRITE-RECORD.
*  Module 7.2 added code
*    MOVE SPACES TO DBG-MESSAGE-ALL.
*    MOVE '  W-RCD ' TO DBG-MSG1 .
*    WRITE FAVOUT-REC FROM DEBUG-REC.
     MOVE FAVIN-REC TO FAVOUT-REC.
     MOVE COMPUTED-COST TO COMPUTED-COST-OUT.
     MOVE ' Cost is:' TO   COST-IS    .
*  end of  Module 7.2 added code
     WRITE FAVOUT-REC.
 CLOSE-FILES.
*    MOVE SPACES TO DBG-MESSAGE-ALL.
*    MOVE ' CLOS-FIL ' TO DBG-MSG1 .
*    WRITE FAVOUT-REC FROM DEBUG-REC.
     CLOSE FAVIN1.
     CLOSE OUT2.
 END-OF-FILE.
     MOVE 'Y' TO LASTREC.
*    MOVE SPACES TO DBG-MESSAGE-ALL.
*    MOVE ' E-O-FILE ' TO DBG-MSG1 .
*    WRITE FAVOUT-REC FROM DEBUG-REC.
 WRITE-LAST-REC.
*    last Record Calculations
     COMPUTE AVREAGE-CD-SALE = COST-TOTAL / REC-KTR .
     MOVE REC-KTR TO REC-KTR-OUT .
     MOVE COST-TOTAL TO COST-TOTAL-OUT .
     MOVE AVREAGE-CD-SALE TO AVREAGE-CD-SALE-OUT .
     WRITE FAVOUT-REC FROM TRAILLER-1.
*     FILL TRAILLER-2
     MOVE  CD-COST-HIGHEST TO  CD-COST-HIGHEST-D  .
     MOVE REC-NO-HIGHEST TO  REC-NO-HIGHEST-D  .
     MOVE  CD-COST-LOWEST TO  CD-COST-LOWEST-D
     MOVE  REC-NO-LOWEST TO  REC-NO-LOWST-D .
     WRITE FAVOUT-REC FROM TRAILLER-2 .
 GET-HIGHEST-LOWEST-CD-COST.
     IF  COMPUTED-COST > CD-COST-HIGHEST THEN
```

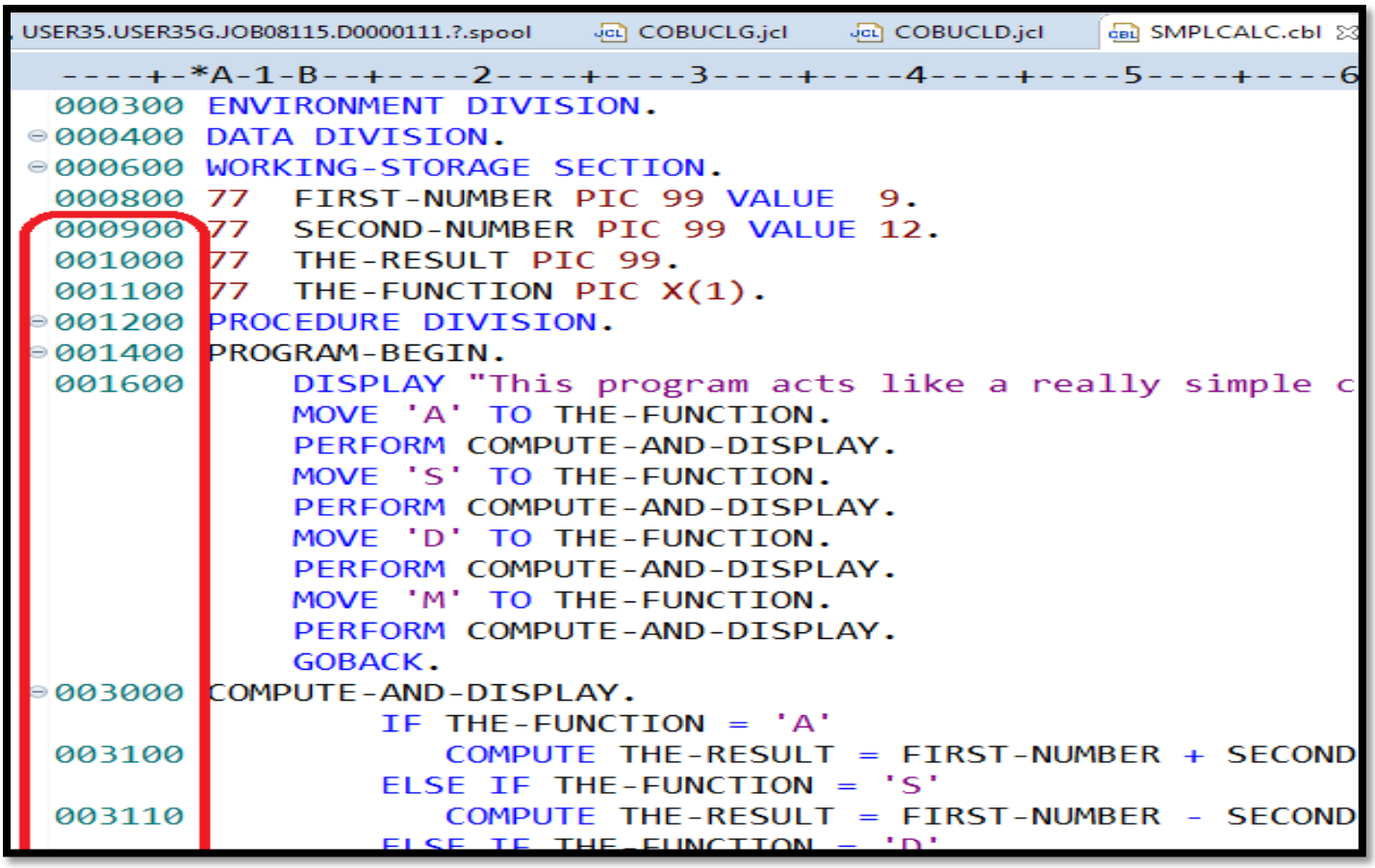
```
            COMPUTE CD-COST-HIGHEST = COMPUTED-COST
            COMPUTE  REC-NO-HIGHEST = REC-KTR
        ELSE IF  COMPUTED-COST < CD-COST-LOWEST   THEN
            COMPUTE CD-COST-LOWEST = COMPUTED-COST
            COMPUTE  REC-NO-LOWEST = REC-KTR    .
```

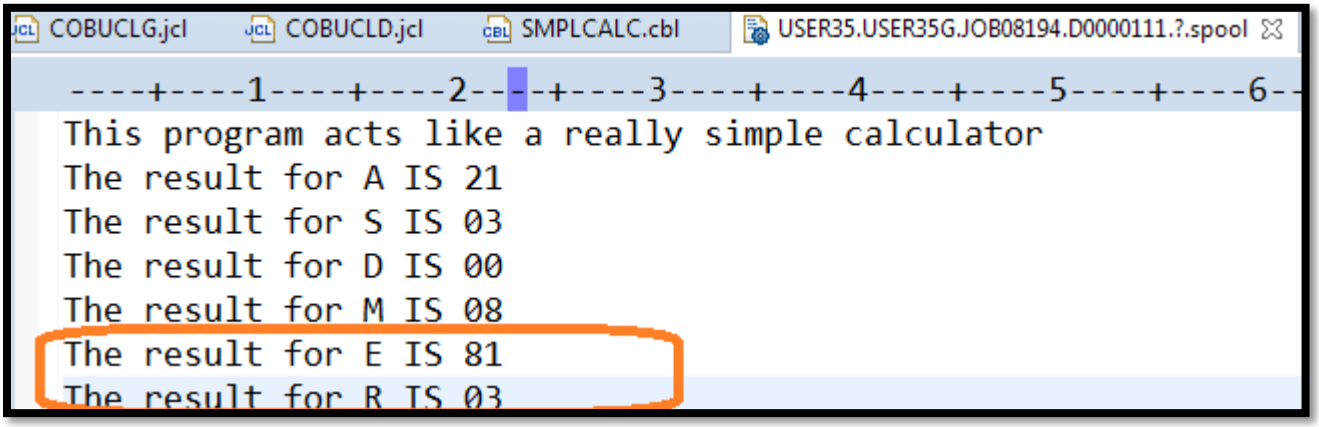
Question 7.3 :



Answer :

Error was there due to the numbers of program lines (were not in order) , removal of error done by reorder .






```
000100 IDENTIFICATION DIVISION.
00000100
000200 PROGRAM-ID. SMPLCALC.
00000200
000300 ENVIRONMENT DIVISION.
00000300
000400 DATA DIVISION.
00000400
000600 WORKING-STORAGE SECTION.
00000500
```

```
000800 77  FIRST-NUMBER PIC 99 VALUE  9.
00000600
000900 77  SECOND-NUMBER PIC 99 VALUE 12.
00000700
001000 77  THE-RESULT PIC 99.
00000800
001100 77  THE-FUNCTION PIC X(1).
            88  ADDTION            VALUE 'A' .
            88  SUBTRACTION        VALUE 'S' .
            88  DIVISION-BY        VALUE 'D' .
            88  MULTIPLICATION     VALUE 'M' .
            88  EXPONENT           VALUE 'E' .
            88  SQURE-ROOT         VALUE 'R' .

001200 PROCEDURE DIVISION.
00001000
001400 PROGRAM-BEGIN.
00001100
001600     DISPLAY "This program acts like a really simple
calculator". 00001200
           MOVE 'A' TO THE-FUNCTION.
00001300
               PERFORM COMPUTE-AND-DISPLAY.
0000
           MOVE 'S' TO THE-FUNCTION.
00001500
               PERFORM COMPUTE-AND-DISPLAY.
0000
           MOVE 'D' TO THE-FUNCTION.
00001700
               PERFORM COMPUTE-AND-DISPLAY.
0000
           MOVE 'M' TO THE-FUNCTION.
00001900
              PERFORM COMPUTE-AND-DISPLAY.
           MOVE 'E' TO THE-FUNCTION.
00001900
               PERFORM COMPUTE-AND-DISPLAY.
           MOVE 'R' TO THE-FUNCTION.
00001900
               PERFORM COMPUTE-AND-DISPLAY.
0000
```

```
           GOBACK.
00002100
003000 COMPUTE-AND-DISPLAY.
00002200
               IF ADDTION
003100            COMPUTE THE-RESULT = FIRST-NUMBER + SECOND-NUMBER
00002400
               ELSE IF  SUBTRACTION
003110            COMPUTE THE-RESULT = FIRST-NUMBER - SECOND-NUMBER
00002600
               ELSE IF  DIVISION-BY
003120            COMPUTE THE-RESULT = FIRST-NUMBER / SECOND-NUMBER
00002800
               ELSE IF MULTIPLICATION
003130            COMPUTE THE-RESULT = FIRST-NUMBER * SECOND-NUMBER
               ELSE IF SQURE-ROOT
003132            COMPUTE THE-RESULT = FIRST-NUMBER ** (0.5)
               ELSE IF EXPONENT
003133            COMPUTE THE-RESULT = FIRST-NUMBER ** SECOND-NUMBER.

003300     DISPLAY "The result for " THE-FUNCTION " IS " THE-RESULT.
00003100
```

Question

Optional Workshop 7.6 - **Arithmetic Precision**

Open the **MORTGAGE** source file, and find the **COMPUTE** statement below.

- What might happen to **MONTHLY-PAYMENT** if **INT-RATE** was defined with a lower precision?

(Don't guess - let's find out):

- Compile/Link and **Debug MORTGAGE**
  - Jot down the value of **MONTHLY-PAYMENT**.
- Duplicate **INT-RATE** in Working-Storage
- Define your new field as: **PIC V9999.**
- Comment out the correct **INT-RATE** definition then Compile/Link and Debug MORTGAGE again.
- What happened? Why?
- Finally, change INT-RATE's definition to: **PIC V99** and repeat the above. What happened now? Why?

```
COMPUTE INT-RATE =
      (03 / 100) / 12.
COMPUTE MONTHLY-PAYMENT
         = PRINCIPAL *
           (INT-RATE *
        (1 + INT-RATE) ** NBR-OF-PAYMENTS) /
           (((1 + INT-RATE ) ** NBR-OF-PAYMENTS) - 1).
```

194

Answer :

COBUCLD.jcl | COBUCLG.jcl | MORTGAGE.cbl | USER35.USER35G.JOB08315.D0000103.?.spool

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7--|-+----8
000016                          (03 / 100) / 12.
000017                       COMPUTE MONTHLY-PAYMENT

==000017==> IGYPG3113-W Truncation of high-order digit positions may occur due to precision o
                        intermediate results exceeding 30 digits.

000018                             = PRINCIPAL *
000019                               (INT-RATE *
000020                            (1 + INT-RATE) ** NBR-OF-PAYMENTS) /
000021                               (((1 + INT-RATE ) ** NBR-OF-PAYMENTS) - 1).
000022                 *
000023                       DISPLAY  'CALCULATED MONTHLY-PAYMENT:' MONTHLY-PAYMENT  .
000024                       MOVE .03 TO INT-RATE.
000025                       COMPUTE MONTHLY-PAYMENT =
000026                       PRINCIPAL * FUNCTION ANNUITY((INT-RATE/12) NBR-OF-PAYMENTS).

==000026==> IGYPS0001-W A blank was missing before character "/" in column 50.  A blank was
                        assumed.

==000026==> IGYPS0001-W A blank was missing before character "1" in column 51.  A blank was
                        assumed.

000027                       DISPLAY  'FUNCTION ANNUITY MONTHLY-PAYMENT:' MONTHLY-PAYMENT.
000028                       GOBACK.
```

conclusions :

1. To avoid truncation the computed data item must be with many decimal positions , also rearrange the compute (make the multiplication operations before the divide operation.
2. if there is an intrinsic Cobol function , Use it , it is ALWAYS accurate . (also it makes the code very readable) .

Question? :

Optional Workshop 7.9 - **Internal Arithmetic Precision**

Create a new version of TESTCOB with the following code ➔

Run or Debug the program and note the differences in results.

```
77  ACCT-VAL-A                PIC S9(15)V9(02) COMP-3.
77  ACCT-VAL-B-01             PIC S9(16)V9(02) VALUE 0.
77  ACCT-VAL-B-02             PIC S9(13)V9(05) VALUE 0.
77  ACCT-VAL-C                PIC S9(16)V9(02) VALUE 0.
77  ACCT-RESULT               PIC S9(15)V9(02) VALUE 0.
.............
.............
     MOVE 2500.87              TO ACCT-VAL-A.
     MOVE 12285                TO ACCT-VAL-B-01.
     MOVE 12285                TO ACCT-VAL-B-02.
     MOVE 4387.5               TO ACCT-VAL-C.
     COMPUTE ACCT-RESULT ROUNDED =
        (ACCT-VAL-A / ( ACCT-VAL-B-01 + ACCT-VAL-C) * 100 ).
     DISPLAY ACCT-RESULT.

     COMPUTE ACCT-RESULT ROUNDED =
      (ACCT-VAL-A / ( ACCT-VAL-B-02 + ACCT-VAL-C) * 100 ).
     DISPLAY ACCT-RESULT.
```

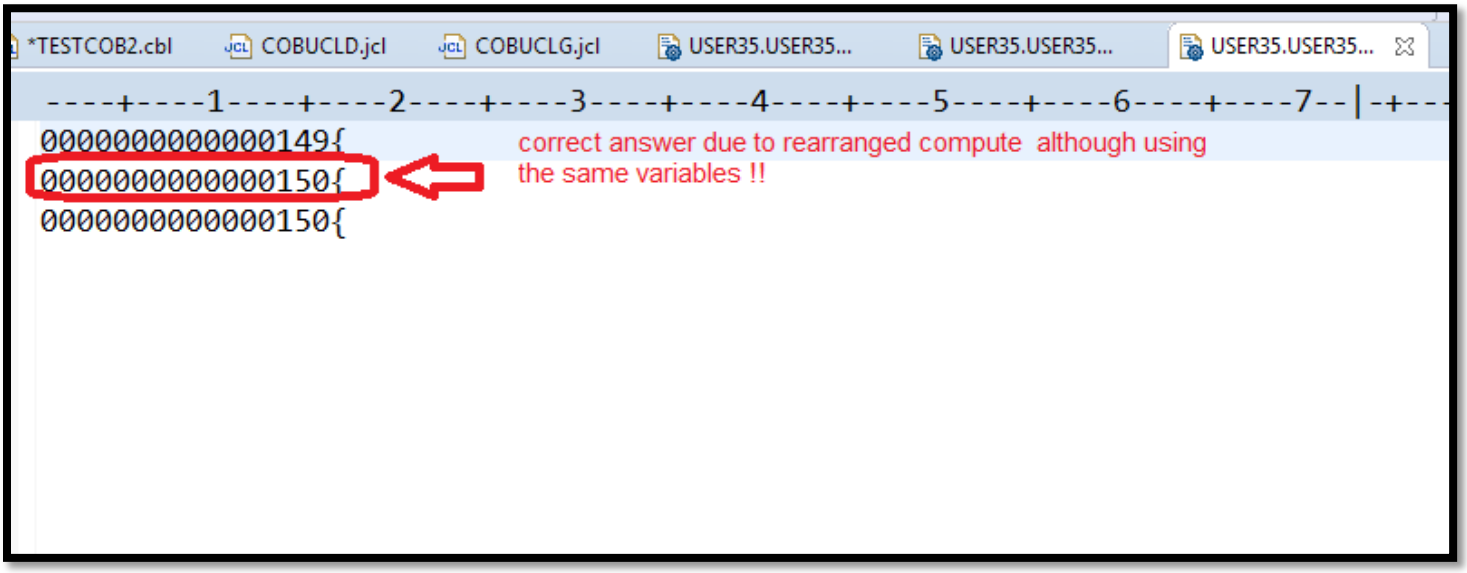
eMail your instructor the reason for these precision issues

196

Answer :

https://stackoverflow.com/questions/25127356/cobol-compute-issues

modified solutiion




```
       IDENTIFICATION DIVISION.
    PROGRAM-ID. TESTCOB2.
```

```
      * Comment: This program Displays a number of text strings
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       77 ACCT-VAL-A          PIC S9(15)V9(02) COMP-3 .

       77 ACCT-VAL-B-01       PIC S9(16)V9(02) VALUE 0.
       77 ACCT-VAL-B-02       PIC S9(13)V9(05) VALUE 0 .
       77 ACCT-VAL-C          PIC S9(16)V9(02) VALUE 0 .
       77 ACCT-RESULT         PIC S9(15)V9(02) VALUE 0 .
       PROCEDURE DIVISION.
           MOVE 2500.87 TO ACCT-VAL-A
           MOVE 12285 TO ACCT-VAL-B-01
           MOVE 12285 TO ACCT-VAL-B-02
           MOVE 4387.5 TO ACCT-VAL-C
           COMPUTE ACCT-RESULT  ROUNDED =
             (ACCT-VAL-A / ( ACCT-VAL-B-01 +  ACCT-VAL-C) * 100 ).
             DISPLAY ACCT-RESULT
      *      The actual problem is a poorly-formed COMPUTE.
      *  TRY TO  Do
      *  multiplication first =>which increase the value
      *  ( or even elemenate ) decimal postions ==> no rounding
      *   and do the division at last ...to just make round one time
      *https://stackoverflow.com/questions/25127356/cobol-compute-
issues
      *   Corrected answer due to rearranged compute
           COMPUTE ACCT-RESULT  ROUNDED =
              ACCT-VAL-A * 100 / ( ACCT-VAL-B-01 +  ACCT-VAL-C)  .
             DISPLAY ACCT-RESULT
           COMPUTE ACCT-RESULT  ROUNDED =
             (ACCT-VAL-A / ( ACCT-VAL-B-02 +  ACCT-VAL-C) * 100 ).
             DISPLAY ACCT-RESULT
           GOBACK.
```