**CSCI 465/565 Assignment 8 Fall 2017  
 External Programs & Linkage  
 150 points**

**Overview**  
    
For Assignment 6, we are going to take the paragraph(s) that we wrote to build the mutual fund table in our Assignment 5 COBOL program and put them in a short new external COBOL program named BUILDTBL.

After we get BUILDTBL fully functional and verified, we will then take the few lines of code used to calculate the number of shares sold in our Assignment 5 COBOL program and put them in a short new Assembler program named CALCSHRS. (If you remember, you have already done these calculations in Assembler in you Assignment 4 program!)

Use the same data sets as those used for Assignment 5. Name your main COBOL "driver" program SALESRPT.

The first external program you are to write will be written in COBOL and will be named BUILDTBL.  It will be called statically.  BUILDTBL will open and read the first two files and build the two-dimensional Fund Table that is defined in the main program.  You will need to pass BUILDTBL the address of the table as a parameter.  Before returning to the main program, set the return code to 0 using COBOL's special register named RETURN-CODE. Only begin the following once you have BUILDTBL working correctly.

The second external program you are to write will be written in Assembler and will be named CALCSHRS.   This program will be called dynamically.  CALCSHRS will compute the number of mutual fund shares based on a deposit and a mutual fund share price.  You will need to pass CALCSHRS the deposit amount, the appropriate mutual fund share price, and the name of the variable into which it will place the calculated share amount.

To make things easier, it is recommended that you pass the values as packed decimal variables to the sub-program.  In the sub-program, do all arithmetic in packed decimal, return the calculated share amount as a packed decimal value, and set the return code to 0 before returning to the main program.  Hint:  You have done this calculating in Assembler packed decimal before!

All other processing will still be done in your main COBOL program, or   
  
SALESRPT.  Your reports should look EXACTLY like they did for Assignment 5.

**JCL Jobstream**

Your jobstream is going to look complicated but we are going to simulate a somewhat realistic external program linkage scenario.  Set your JCL up as follows:

1.  COBOL Compiler for SALESRPT.  
2.  COBOL Compiler for BUILDTBL.   
3.  Linkage Editor for the two object modules created in steps 1 and   
 2 above which creates a load module containing both SALESRPT and   
 BUILDTBL because BUILDTBL is being called statically(!).  This   
 load module MUST have the same name as the main calling program.   
4.  Assembler for CALCSHRS.  
5.  Linkage Editor for CALCSHRS which creates a load module of   
 CALCSHRS only and adds this to your load library PDS.  
6.  DFSORT of the input mutual fund data like that necessary in   
 Assignment 5 for DATA5A.  
7.  Fetch and execute SALESRPT.  Remember that you will have to use a  
 STEPLIB here to reference the load library where both your main   
 driver program, SALESRPT's and the load module for the   
 dynamically called program CALCSHRS are stored.

**Programming Notes**

You **might** need to use the Linkage Editor ENTRY control statement in step 3 of the above to identify to the Linkage Editor which of the two object modules is to be considered the entry point (i.e., the first program to be executed).  This can be done as follows:

Linkage Editor control statements are included in SYSLIN as an in-stream data set concatenated with the object modules, as in:

//SYSLIN DD DSN=&&OBJ1,UNIT=PUB,DISP=(OLD,DELETE)  
 // DD DSN=&&OBJ2,UNIT=PUB,DISP=(OLD,DELETE)  
 // DD \*  
 ENTRY SALESRPT  
 MODE AMODE(24)  
 /\*

Do NOT link edit ANYTHING or even proceed to the next compile or assembly step unless ALL previous steps are completely error free!  
    
As in Assignment 5, use the COPYLIB members at least once each possible but use KC02322.CSCI465.COPYLIB.

You should pass the 'FLAG(I,I),APOST,LIB' options to the COBOL compiler both times.

You should pass the 'NOESD,NORLD,NOXREF, ASA' options to the Assembler.

You should pass the MAP option to the Linkage Editor both times.  
    
Include facilities to help you debug this program.  These are described in your course notes.

You may want to use TERMTHDACT(DUMP).  Remember that if a dump is generated, it is written to a DD statement:

//CEEDUMP DD SYSOUT=\*

For more simplicity's sake, just write the three programs instream.  In order to get this to work as you develop it, though, you will have to use the JCL jobstream described above.

Submit your ***single***textfile for grading.