**CSCI 465/565 Assignment 6 – QSAM Fall 2017**

**250 points**

**Overview**

This assignment's objective is to give you experience using the five Assembler QSAM macros and hone your Assembler programming skills.  You will create the same reports you did for Assignment 5.

Utilize the file DATA5 the same way you did in Assignment 5. Note that the share price may change periodically.

Write an Assembler QSAM program that creates the same two well-designed printed reports as the COBOL program you wrote for Assignment 5. Read and follow the Programming Notes below VERY carefully.

**Programming Notes:**

* Name your ASSIGNS PDSE member for this assignment ASSIGN6 and only ASSIGN6 so that Mr. Decker can review your progress at any point.
* Be sure to compress your ASSIGNS PDSE periodically with Option 3.1 in ISPF.
* Be sure to back up your ASSIGNS PDSE periodically; if it is corrupted or you accidentally delete it, it cannot be restored.
* Write your Assembler QSAM program using GM and PM macro formats first. Once you get it working perfectly, then change to the macro formats in the following two bullet points.
* Use the GM version of the GET macro for accessing DATA5.
* Use the PM and GL versions of the PUT and GET macros for accessing the temporary data set for high sales.
* Use the PL version of the PUT macro for writing records to the output for your two reports.
* No register equates allowed.
* Do not use literals except for numeric-edited output edit patterns. Note that long numeric-edited output edit patterns should be defined in storage instead of using a literal.
* Use plenty of wisely-placed asterisks in column 1 to spread your code out vertically for easier reading.
* Macro names, DSs, DCs, mnemonics, etc., all go in column 10. Operands and storage definitions begin in column 16 (Note that the names of ***some*** IBM macros are so long that they require parameters start in a column following 16. If so, that’s okay).
* Use a PRINT NOGEN above your Assembler program doc box to suppress macro expansion in your source listing.
* Immediately following the required LTORG, use the 32-byte boundary ORG technique to easily identify your storage in a dump.
* You may use DSECTs. Define all of your DSECTs above the CSECT. All DSECT names and field names must begin with a dollar sign ($).
* Remember that the temporary data set will only need a single DCB.
* Test each file OPEN for success using LTR 15,15. If the return code is something other than 0, force an abend using the ABEND macro with a different USER code for each OPEN.
* Do **all** arithmetic in packed decimal. Your line counter can be a register, though.
* Do **NOT** use MVC to move packed decimal numbers. Use ZAP instead **but only move them if necessary**. The only times you should have to ZAP packed decimal numbers should be 1) to prepare the deposit for shifting (SRP) and divide packed (DP) and possibly to prepare the deposit for multiply packed (MP) to calculate commissions.
* Use the TIME macro **ONLY** once.
* Declare DCBs at the end of your storage with any required EODAD EOF routines and EOF flags declared **immediately following** the DCB definition for that file.
* Use good line documentation and other documentation as described in the CSCI 465/565 Course Notes.
* Be sure that ALL 133-byte output line definitions have spaces defined in between the fields that will receive values.
* Do NOT use subroutines of any type.
* You may NOT use XSAVE and XRETURN for standard linkage. Use that provided below:

Standard Entry Linkage

MAIN     CSECT                                                          
         STM   14,12,12(13)    SAVE CALLER'S REGS                     
         LR    12,15           SET R12 TO R15                         
         USING MAIN,12         ESTABLISH R12 AS 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  
    
Standard Exit Linkage with RC = 0 in R15  
   
         SR 15,15 RC = 0

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

* Use the advanced Assembler techniques discussed in class. For example, to put a value between 1 and 4095 into a register, use LA, to set a register to 0, subtract the register from itself, to decrement a register by 1, use BCTR, etc.

As before, what you will hand in will be one single job .txt file with an instream Assembler program followed by a Linkage Editor step followed by a fetch and execute step similar to what you have done before.

Note that, if you were to print the reports from Assignment 5 and those produced by this assignment’s Assembler QSAM program, you should be able to hold them up to the light and see only differences in the date and time.

**What to Turn In**

Use mar\_ftp.exe to get the file of the output down from the Marist output queue onto your PC.  Submit as directed.