CSCI 465 **Assignment 7 – COBOL Tables and Subprograms** **Spring 2020**
250 points

Overview

Because this is one of your final mainframe projects before graduating, it is important that you exhibit as high a degree of professional-level mainframe programming as is possible at this point in your education. The quality and efficiency of your programming will be a large part of your grade. As Mr. Decker will be grading this assignment, please pay **VERY** close attention to the ***Coding Guidelines*** provided near the end of this handout and any made previously available for other assignments this semester.

AS MR. DECKER WANTS TO REVIEW YOUR WORK AS YOU COMPLETE THIS ASSIGNMENT, ALL OF YOUR PROGRAMMING MUST BE DONE IN EITHER TSO/ISPF OR RATIONAL DEVELOPER FOR z SYSTEMS. IF IT IS DISCOVERED THAT YOU ARE USING A DIFFERENT PLATFORM, YOU WILL EARN A ZERO ON THIS ASSIGNMENT.

In this assignment we are using single and multi-dimensional tables to process a business day's worth of broker deposit and sales data and create a single report for the Feelings Mutual Fund Company. Up to this point, we have only considered one mutual fund but this new company manages many more funds than just one.

Each business day, each broker is able to sell mutual fund shares from any of those offered by the firm. The number of shares of each mutual fund sold by a broker each business day depends on the closing price of each share of each mutual fund sold and, of course, the deposit (in dollars and cents) for that mutual fund sold by a broker during that business day.

In addition to table processing, this project will involve one main COBOL program that calls two external subprograms, one of them written in COBOL and the other written in Assembler. The main COBOL program will process the main sales member named `KC02322.CSCI465.DATA7(SALES)`. The COBOL subprogram will read the file named `DATA7` in this semester's data PDSE to build the mutual fund table. The Assembler subprogram will calculate both commission and share amount for each of a broker's up to four sales in a single day.

The data set member and sequential data set represent the following:

1) `KC02322.CSCI465.DATASP20(DATA7)` - The number and name of each mutual fund managed by the firm, the price per share of that mutual fund and the percent commission paid to the broker for selling shares of its fund.
2) `KC02322.CSCI465.DATA7(SALES)` - A PDSE member of broker daily deposits and sales.

The PDSE member, DATA7, is unsorted. Before you begin executing your COBOL program, you will need to sort the data set using a JCL EXEC step that runs DFSORT (see example). Sort it in ascending order on the mutual fund number (first three bytes, zoned decimal). Write the sorted data to a temporary data set which you will pass to the executing step of your program and use as input instead of member DATA7 itself.

Open and read the first file from beginning to end and fill, or "build", the COBOL table. The table needs to be a ***variable length table*** using an OCCURS DEPENDING ON phrase. Note that the DEPENDING ON variable's PIC clause should include a sign and should be a binary sync. The field ***must be*** initialized to 0 to start.

After the table is "built", you can open and begin processing the sequential file numbered 2) above from beginning to end and create the "DAILY FUND SALES REPORT".

The Fund Table

Each entry of the fund table will hold the mutual fund number, the name of the mutual fund, the price per share of the mutual fund and the commission percent paid to the broker for selling shares of the mutual fund. Already sorted by mutual fund number, each record from the this input file will provide the mutual fund information in the following format:

```
01  IN-FUND-RECORD.
    05  IN-FUND-NBR         PIC 9(3).
    05  IN-FUND-NME         PIC X(25).
    05  IN-FUND-SHR-PRC     PIC S9(3)V99.
    05  IN-FUND-COMM-PCT    PIC S9V9(5).
```

(The above was taken directly from the COPYLIB member FUNDREC.)

The fund number is an integer value between 1 and 999 and there CAN be up to 999 mutual funds. It must be indexed and, after being built, will be in ascending order based on the fund number.

The fund table that you define in your WORKING-STORAGE SECTION will be ***partly*** defined by the following:

```
        10  TBL-FUND-NBR         PIC S9(3)    BINARY SYNC.
        10  TBL-FUND-NME         PIC X(25).
        10  TBL-FUND-SHR-PRC     PIC S9(3)V99 BINARY SYNC.
        10  TBL-FUND-COMM-PCT    PIC S9V9(5)  BINARY SYNC.
```

(The above was taken directly from the COPYLIB member FUNDTBL.)

You will need to define the 01- and 05-levels of the table yourself but the above lines are to be "plugged in" using the COPY statement.

The Sales File

Each record of the sales file represents a one-dimensional table. This means that there will be a one-dimensional table definition under the FD for this file.

At the beginning of each sales record are the city name and broker name. The following is the sales record format:

```
01  SALES-RECORD.
    05  SALES-BRANCH-NME          PIC X(25).
    05  SALES-BROKER-NME          PIC X(25).
    05  SALES-FUND-SALE           OCCURS 4
                                  INDEXED BY SALE-NDX.
        10  SALE-FUND-NBR         PIC S9(3).
        10  SALE-DEP-AMT          PIC S9(8)V99.
```

(The above was taken directly from the COPYLIB member SALESREC.)

Following the broker name is where the one-dimensional table that is built into the input sales record is found. Although a bit unrealistic but done to simplify things, a broker can only make up to four different mutual fund sales per day. (Note that there may be repeated mutual funds in these up to four sales!)

Along with the city name and broker name, adding up to 50 bytes, these four 13-byte sets of fund sales information make up 102 bytes total. Note that one or more of these four fund sales sets of data may contain all zeros if the broker sells fewer than four mutual funds during the day. If the fund number is set to zoned decimal zeros, you can ignore the rest of the data on the record and move on to the next sales record in the file.

For each record representing a broker's sales for the business day, you must first move his or her city name and broker name to the detail line of the DAILY FUND SALES REPORT.

You will then parse each of the four 13-byte fund sales sets of data. For each that has a fund number greater than 0, do a binary search (COBOL's SEARCH ALL) using the sales record's fund number as the search key into the Fund Table. If a match is found, move the fund number and the fund name from the Fund Table to the detail line following the broker's name.

Note that the broker's city, and name should only appear on the first detail line for his or her first, or perhaps only, sale of the day.

Then, move the deposit amount to the detail line. Then, move the calculated number of shares to the detail line. Finally move the calculated commission amount paid to the broker for this particular sale.

Note that there is also NO high earner processing in this program.

The first detail line for each broker will have the following information from left to right: branch name, broker name, fund number, fund name, deposit amount, share amount and commission amount (for just that sale). If the first fund number is not found in the table, the fund number will be followed by the not found message, and the deposit amount with a $0.00 in the commission amount field.

Note: Your detail line is going to be tight. The exact output provided is probably the best way to set up your report headers, column headers, hyphens and detail lines.

The possible second, third and fourth partial detail lines for each broker will have only the following information from left to right: fund number, fund name, deposit amount, share amount and commission amount (for just that sale). If any of the second, third, and/or fourth fund numbers are not found in the table, the fund number will be followed by the fund not found message, the deposit amount and zeroes in both the share amount and commission amount fields.

Count the number of brokers reported, count the number of sales (including those with no matching fund found), accumulate the grand total of all of the deposits and accumulate the grand total of all of the commissions to be paid as you process the sales file.

Do not worry about calculating and reporting the number of shares processed but you WILL be calculating the average total deposit across all brokers. This means that you need to add each broker's deposits to a grand running total across all sales for all brokers. At the end you will divide this total by the number of brokers.

(continued)

The COBOL TBLBUILD Subprogram

You will declare the Fund Table in the main program named FNDSALES and you will pass the address of the Fund Table to a COBOL Subprogram named TBLBUILD that you will call dynamically. TBLBUILD will open the file sorted from DATA7, fill the table with the mutual fund data and will then return to FNDSALES. DO NOT BUILD THE FUND TABLE IN FNDSALES!

The Assembler CALCFUND Subprogram

As you process each broker sale of some number of shares of a mutual fund, you will search for a match of the mutual number number in the Fund Table. When you find a match, you will then call an Assembler subprogram named CALCFUND statically. Pass this program the addresses of five fields: deposit amount of the current sale, share price of the mutual fund found in the Fund Table, a field into which CALCFUND will place the share amount calculated, the commission percent of the mutual fund found in the Fund Table and, finally, a field into which CALCFUND will place the commission amount calculated. Once these calculations are complete, CALCFUND will then return to FNDSALES so that the share amount calculated and commission amount calculated can move the amounts to the detail line and write it.

JCL Jobstream

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

1. DFSORT of the input mutual fund data in PDSE member DATA7.
2. COBOL Compiler for FNDSALES.
3. Assembler for CALCFUND.
4. Binder (Linkage Editor) for the two object modules created in steps 2 and 3 above which creates a load module containing both FNDSALES and CALCFUND because CALCFUND is being called statically(!). The output program object of this step MUST have the same name as the main calling program, FNDSALES.
5. COBOL Compiler for TBLBUILD.
6. Binder for TBLBUILD which creates a program object of TBLBUILD only and adds this to your load library PDSE.
7. Fetch and execute FNDSALES. Remember that you will have to use a STEPLIB here to reference the load library PDSE where both your main driver program, FNDSALES's and the program module for the dynamically called program TBLBUILD are stored.

Details About the Report

1. Center the titles under one another at the top of each page, including the totals pages. This should include the name of the mutual fund company in the first header and "DAILY FUND SALES REPORT" as a subtitle in the second header.
2. Each page should have the date in MM/DD/YYYY format at the far left of the first header line and 'PAGE: ' with page number in format ZZ9 at the far right of the first header line.
3. Each page should have the time in HH:MM:SS format at the far left of the second header line.
4. Double-spaced after the second of the page headers, or titles, should be an appropriate column header (which can be two lines if necessary) and, single-spaced after that, a line of appropriately placed hyphens.
5. Maximum of 17 double-spaced broker detail lines per page. It is permissible to page break while listing a broker's four possible sales for the day.
6. Use the full 132 bytes available to spread out your headers and detail lines.

7. All arithmetic calculations for dollars and cents should be rounded to two decimal places. All calculations for share amounts should be rounded to three decimal places with a fourth decimal place set to 0 when displayed.
8. All numeric-edited fields should be correctly edited with a floating dollar sign (when dollars and cents), commas between the thousands, and decimal points.
9. Remember to ONLY print the broker's city name, and broker name on the first of the four possible detail lines reporting his or her daily sale(s).
10. If the binary search for a matching mutual fund in the Fund Table returns a not found condition, move the numeric-edited mutual fund number (the one that's not been found in the table, that is) to the fund number field in the detail line and the following 25-byte text to the ***fund name field*** in the detail line:

    `*** FUND NBR NOT FOUND **`

    Finally, move the deposit amount to the detail line following this not found message. Note that you SHOULD also add the deposit amount to the grand total of deposits in spite of not being able to find a matching fund. Move 0 to both the share amount and commission amount columns.
11. At the end of this report and at the top of a new page, double space and center a ***third*** title (second subtitle, that is) of "`****** TOTALS ******`". Double-spaced after this third header, center your column headers and, single-spaced after that, a line of appropriately placed hyphens. Double-spaced below this display the number of brokers, the total of all deposits rounded, of course, to two decimal places, and the average deposit per broker reported rounded, of course, to two decimal places, and then the total of all commissions rounded, of course, to two decimal places. Remember that the average of the deposits is all of the broker deposits added to a single grand total and divided by the number of brokers counted.

Coding Guidelines

1. No printing of headers prior to the loop processing the sales records.
2. No COBOL arithmetic verbs allowed unless adding 1 to a broker counter, page counter, sales counter, subscript, etc.
3. Only COMPUTE ROUNDED allowed for arithmetic calculations.
4. ALL WRITE statements must include either AFTER 1, AFTER 2, or AFTER PAGE.
5. NO printing blank lines allowed.
6. Numeric fields that will be printed must be declared as PACKED-DECIMAL.
7. Numeric fields that are only used for arithmetic and will never be printed must be declared BINARY SYNC.
8. Fields declared as indexes must end with -NDX and those declared as subscripts must end with -SUB.
9. Fields declared as subscripts must be declared as BINARY SYNC.
10. The headers for the main report should appear within the FNDSALES read loop and nowhere else.
11. The headers for the totals page should appear after FNDSALES read loop and nowehere else.
12. There can be up to 999 brokers and sales so declare the broker and sales counters appropriately.
13. Move 0 to the COBOL special register RETURN-CODE right before the GOBACK in both COBOL programs.
14. In the Assembler program, set register 15 to zero as a return code and do not restore register 15 in the exit linkage.
15. No extra, unnecessary and/or unused variables allowed.
16. Inefficient coding in any of the three programs will be penalized appropriately.

(continued)

Other Notes

It is required that you use each of the COPYLIB members FUNDTBL, FUNDREC, and SALESREC in copy library, or "copylib", PDSE named KC02322.CSCI465.COPYLIB in your program.

Send the data to the Assembler subprogram CALCFUND in packed decimal format and do all of the arithmetic in CALCFUND in packed decimal. Return the data in packed decimal.

Note that you might have to add some filler after the lines of code copied in from the COPYLIB members to equal 80 bytes for any of them related to input records for DATA7 and 102 bytes for any of them related to input records for SALES.

DFSORT is necessary for this program for member DATA7 (Make it the first step of your job).

Call the COBOL intrinsic date function only once in your program and NOT inside of a loop.

Do NOT put a long string of COBOL commands within any part of the SEARCH ALL binary search structure. You can PERFORM a new paragraph or two, if necessary, at that point.

Be careful to follow the guidelines above and the guidelines in the course notes book about COBOL, Assembler and JCL documentation and submit the .txt file with the above four steps on Blackboard.