

**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 possible at this point in your education. The quality and efficiency of your programming will be a large part of your grade. 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 IBM DEVELOPER FOR z SYSTEMS (IDz). 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 sale data and create a single report for the company Washington & Lincoln Investments which manages more funds than just the one with which we have worked in previous assignments.

Each business day, each of Washington & Lincoln Investments' brokers 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 the two subprograms will be written in COBOL and the other written in Assembler. The main COBOL driver program, SALERPT, will process the main sale file named KC02322.CSCI465.DATA7(SALES). The COBOL subprogram, BUILDTBL, will read the file named FUNDS in this semester's data PDSE to only just build the mutual fund table. The Assembler subprogram, CALCAMTS, will calculate both share amount and commission for each of a broker's up to four sale in a single day.

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

- 1) KC02322.CSCI465.DATAFA21(FUNDS) - 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 sale.

The PDSE member, FUNDS, 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 the data set 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 FUNDS itself.

In program SALERPT, 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 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 BROKER DEPOSIT 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.

To "fill" this table, each record in the input file – already sorted by mutual fund number – each record from 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 S9(3)V9(3) .
```

The above was taken directly from the COPYLIB member FUNDREC. The fund number is an integer value between 1 and 199 and there CAN be up to 199 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)    PACKED-DECIMAL .
10  TBL-FUND-NME          PIC X(25) .
10  TBL-FUND-SHR-PRC      PIC S9(3)V99 PACKED-DECIMAL .
10  TBL-FUND-COMM-PCT     PIC S9V9(5)  PACKED-DECIMAL .
```

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 Sale File

Each record of the sale file contains 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 sale record are the city name and broker name. The following is the sale record format:

```
01  SALE-RECORD.
    05  SALE-BRANCH-NME          PIC X(25) .
    05  SALE-BROKER-NME          PIC X(25) .
    05  SALE-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 SALEREC. Following the broker name is where the one-dimensional table that is built into the input sale record is found. Although a bit unrealistic

but done to simplify things, a broker can only make up to four different mutual fund sale per day. (Note that there may be repeated mutual funds in these up to four sale!)

Along with the city, or branch, name and broker name, adding up to 50 bytes, these four 13-byte sets of fund sale information make up 102 bytes total. Note that one or more of these four fund sale sets of data may contain all zeros if the broker makes fewer than four mutual fund sales 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 sale record in the file.

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

You will then parse each of the four 13-byte fund sale sets of data. For each that has a fund number greater than 0, do a binary search (COBOL's SEARCH ALL) using the sale 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, or branch, name, and the broker's 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 and calculated number of shares to the detail line. Finally, move the calculated commission amount paid to the broker for this particular sale to the detail line. Remember that each mutual fund has its own commission percentage stored in the Fund Table.

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 (see #10 under Details About the Report below) and then the deposit amount followed by a \$0.00 in both the share amount and commission amount fields.

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, just like the first line, 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 sale 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 High Sale File

As before, you will write a record to a temporary data set for each of the up to four sales that a broker has sold more than 10,000.0000 shares of a particular mutual fund. The following is the High Sales Record format:

```
01  HIGH-SALE-RECORD .
    05  HS-BROKER-NME          PIC  X(25) .
    05  HS-FUND-NBR            PIC  S9(3)          PACKED-DECIMAL .
    05  HS-SHR-AMT             PIC  S9(10)V9(3)    PACKED-DECIMAL .
```

The above was taken directly from the COPYLIB member HSREC.

### The COBOL BUILD\_TBL Subprogram

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

### The Assembler CALCAMTS 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 CALCAMTS 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 CALCAMTS will place the share amount calculated, the commission percent of the mutual fund found in the Fund Table and, finally, a field into which CALCAMTS will place the commission amount calculated. Once these calculations are complete, CALCAMTS will then return to SALERPT 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 SALERPT.
3. Assembler for CALCAMTS.
4. Binder (Linkage Editor) for the two object modules created in steps 2 and 3 above which creates a load module containing both SALERPT and CALCAMTS because CALCAMTS is being called statically(!). The output program object of this step MUST have the same name as the main calling program, SALERPT.
5. COBOL Compiler for BUILD\_TBL.
6. Binder for BUILD\_TBL which creates a program object of BUILD\_TBL only and adds this to your load library PDSE.
7. Fetch and execute SALERPT. Remember that you will have to use a STEPLIB here to reference the load library PDSE where both your main driver program, SALERPT's and the program module for the dynamically called program BUILD\_TBL are stored.

### Details About the First 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 investment company in the first header and "DAILY BROKER DEPOSIT 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 with AM/PM 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 sale 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 four 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 **
```

Then 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 "\*\*\*\*\* REPORT 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.

### Details About the Second Report

This report is pretty much the same as previous high sales reports but now you will have up to four broker names repeated and you will have to add the fund number to the detail line. Refer to the exact output for formatting and headers. For the totals, only accumulate and report the number of *brokers* (not records) in the high sales file. Print 17 detail lines per page. Subtitle this report "BROKER HIGH SALES REPORT".

### Coding Guidelines

1. No printing of headers prior to the loop processing the sale records.
2. No COBOL arithmetic verbs allowed unless adding 1 to a broker counter, page counter, sale 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. Names of fields declared as indexes must end with -NDX and names of 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 SALERPT read loop and nowhere else.
11. The headers for the totals page should appear after SALERPT read loop and nowhere else.
12. There can be up to 999 brokers and sale so declare the broker and sale 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 four programs will be penalized appropriately.

### Other Notes

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

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