Project 2: Boston Housing Model Evaluation Program

(300 points)

You are a mainframe developer working for a nation-wide corporation contracted by the City of Boston's Civil Planning and Management services to develop and manage parts of the city's IT infrastructure dedicated to urban development planning. Your division has been tasked with developing a business intelligence module for the system, and you have been given the task of writing the code for a submodule that evaluates a set of linear models generated by the core components of the intelligence software as new data becomes available. The models are represented by sets of coefficients that can be assessed on the original housing data to get an idea of how well they can be expected to generate accurate predictions on unseen data.

The program you will write will take a set of models (a table of coefficients), generate a set of predictions using the housing data with each model, and then calculate the Mean Squared Error (MSE) statistic for those models.

Program Outline

Input

The first record in the input contains the column labels of the housing data. You will write an external subprogram `READLBLS` to read and return them to the main program, using a pointer to an array passed by the main program.

Your program will then read in two sets of data from file. The first is a table of coefficients, where each row of coefficients describes a linear model that your program will evaluate.

The second is the table of housing data that your program will need to calculate the MSE statistic for each model. The models and housing data read from file should be stored in separate tables in memory which you'll declare in your main program storage. Use external subprograms to build the tables, `READMDLS` and `READDATA`.

In both the model and housing data tables read from file, a single row is too big to fit into an 80 byte record so rows have been split across two records each. This means you'll need to perform two `XREAD`s per row. The model and housing data tables are separated by a single asterisk ('*') on its own line, so when your program sees the asterisk it should switch over from reading in the models to reading in the housing data.

The in-memory tables will need to be built to the specifications given in the Table Formatting Specifications subsection under Requirements. Take a look before reading on.

Processing

Once you've built both tables, your main program will need to iterate through the rows of the coefficients table and evaluate each model (i.e., the coefficients in the row) using the entire housing data table. You will write an external subprogram `EVALMODL` which your program will call to perform the computations and return an MSE statistic. The main program should pass `EVALMODL` the address of the model table row and the start address of the housing data table. Once it has finished, `EVALMODL` should then return the MSE in the last column of the coefficients table row (refer to the Table Formatting Specifications subsection).

The `EVALMODL` subprogram will need to do several things. First, it will need to iterate through the housing data table, and for each row (referred to as a *sample*) it should call an external subprogram `CALCYHAT` that will use the model coefficients and the row data to calculate a predicted value for the sample (row) using the following formula:

$$\hat{y} = \sum_{i=1}^{n} c_i x_i \tag{1}$$

where $\hat{y}$ is the predicted value and c_i and x_i are a coefficient and an independent variable value respectively, for the corresponding columns at position i in both the row of coefficients passed to `EVALMODL` and a row in the housing data table. You might notice that there are 15 columns in the housing data table specification but only 13 coefficients in the models that are read from file. The 14th column contains the observations, or known target values of y, which will be used later during model evaluation when you calculate the MSE.

Use an inner loop to iterate through the 13 coefficient-column value pairs. `EVALMODL` will need to pass the address of the row of the housing data table, which also contains a column for the $\hat{y}$'s (see the Table Formatting Specifications subsection for the layout) to `CALCYHAT`, which should put the calculated $\hat{y}$ into the 15th column (refer to the Table Formatting Specifications section).

After the $\hat{y}$'s have been calculated the `EVALMODL` subprogram will need to generate the MSE statistic using the y and $\hat{y}$ values (both of which should be sitting in the housing data table's last two columns). You will write an additional external subprogram that will be called by `EVALMODL`, `CALCMSE`, to handle the computations. `EVALMODL` should pass the address of the MSE column in the row of the model table it was passed as well as the housing data table to `CALCMSE`, which will put the result (the MSE statistic) into the MSE column.

Output

Finally, as there is no other program to call your own program—and even if you were doing this for money, you wouldn't want to plug your program into a live production system for testing anyway—you will need to write an external subprogram `SHWRESLT` to generate a report. The external subprogram should first print the column labels and then iterate through the table containing the models and MSE statistics returned by `EVALMODL`. For each it should print the coefficients and then report the MSE on the same line, using a tabular format.

Requirements and Specifications

External Subprograms

You must implement *all* of the subprograms discussed in the program outline. Beginning in your main program body, you should use the following external subprogram call hierarchy:

- `READLBLS`
- `READMDLS`
- `READDATA`
- `EVALMODL`
  - `CALCYHAT`
  - `CALCMSE`
- `SHWRESLT`

Input Specifications

Column labels are split across two records, so you will need to perform a read twice to get them all. Every label occupies 9 bytes and the labels are consecutive. Labels are pre-formatted for

printing later so you'll want to put all 9 bytes into each position in the array. The array passed to `READLBLS` should have enough space to hold exactly 13 9-byte labels.

Both the following coefficients and housing data are formatted as floating point decimals, which cannot be packed directly because of the embedded decimal character (EBCDIC '4B'). You will use the provided PARSEDEC subprogram as discussed in class (provided to you in the project JCL skeleton) to retrieve the packed decimal values your program will need. Be sure to thoroughly go through the documentation box for usage instructions and the parameter list specifications.

The model coefficients read from file may be positive or negative. After packing a coefficient, your code must then test the sign character trailing each, either a '+' or '-'. For example:

```
0.28513- 0.02682+ 0.01965+
```

You will need to flip the sign of the packed values which should be negative. All coefficient value strings read from file will require *5 implied decimal places*.

The housing data contains only positive quantities, so you will *not* need to worry about processing a sign as was required to handle the coefficients. All values have *4 decimal places*. Recall that a single row of data in both tables is split across two records. For example, in

```
0.0063 18.0000 2.3100 0.0000 0.5380 6.5750 65.2000 4.0900 1.0000 296.0000
15.3000 396.9000 4.9800 24.0000
```

both lines (records) represent a single row in the housing data table, with the 14 values corresponding to each of the 14 columns. Please note that only 13 of these columns represent dependent (i.e., input) variable data; the 14th column contains a target or "ground truth" value that the linear equation—using the coefficients and column data should—should ideally produce (the desired y, against which the calculated value $\hat{y}$ will be compared).

Table Formatting Specifications

Model Coefficients	MSE
13×5 bytes	6 bytes

Table 1: Model Table, Byte Allocation per Row

Housing Feature Data	y	$\hat{y}$
13×5 bytes	5 bytes	5 bytes

Table 2: Housing Data Table, Byte Allocation per Row

You will need to declare storage for both the model and housing data tables to accommodate rows that hold additional columns. These extra columns will store the results of the calculations made while the program executes. Both tables should be declared using packed decimals.

The exact row specifications are given in the tables above. Each row in the model table contains the coefficients for a model read from file plus a column for the MSE statistic to be filled in by the `CALCMSE` subprogram called from `EVALMODL`.

Each row of the housing data table contains the 14 columns of data read from file, including the feature data and the y, plus a column for the $\hat{y}$'s calculated by the `CALCYHAT` subprogram. Note that the values in the $\hat{y}$ column will be different for each model, and each time a model is evaluated the $\hat{y}$'s will change. Note that they're not used outside of `EVALMODL`, but we store them in the table anyway for the sake of streamlining our code (i.e., convenience).

There will be five models to evaluate, so you will need to declare enough storage to hold five rows of the total size given by the sum of the specifications given in Table 1. The housing data table will need to contain 506 rows of the total size given by the sum of the specifications given in

Table 2. That means you'll need to allocate 37,950 bytes to hold all the data. You won't be able to declare it all with a single line of code, so you'll need to break the declaration up into several statements. It doesn't matter how you break it down as long as the total is *at least* 37,950 bytes.

DSECTs

You will need to declare and use `DSECT`'s to establish addressability on both tables in storage. It isn't practical to have a separate label for each coefficient column in the model or feature column in the housing data tables, but you *should* use labels for the MSE, y and $\hat{y}$ columns. For example, in the model table `DSECT` you would have two labels, one for the start of the coefficients and one for the MSE:

```
$MODLTBL   DSECT
$COEFFS    DS     13PL5      13 COLUMNS OF 5 BYTES
$MSE       DS     PL6
```

You will need to write something similar to handle the housing data table. Refer to the table specifications above for byte allocation details. Note that in this example, the `$COEFFS` field contains all 13 columns of the coefficient data; you may at your own discretion create a separate field for each column, however it is not necessary as you will want to iterate through the coefficients when building or reading rows in the table.

Calculating the MSE Statistic

The `CALCMSE` subprogram should use the following formula to calculate the mean squared error (MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{2}$$

where n is the total number of rows in the housing data table. The i denotes the number of a row in the table. There are a few steps involved, but the calculation is not as ugly as it may seem at first glance.

The first thing you'll need is to count the total number of rows that have been read into the housing data table for your n. This is something easily done in the `EVALMODL` subprogram as it calls `CALCYHAT` on each row. You'd then simply pass the address of the count field to the `CALCMSE` subprogram as a parameter.

The summation may look nasty but all it really means is that you're adding things together. For each row in the housing table (you'll need to have passed its address to the `CALCMSE` subprogram) you'll calculate the difference between y and $\hat{y}$, square it, and then accumulate the result into a total. After multiplying the total by $\frac{1}{n}$ the result (MSE) will always be a positive quantity.

Closing Remarks

Standard fare:

- Fully document your program as instructed in the **CSCI 360 Coding and Documentation Guidelines** document on Blackboard.
- Be sure that your *entire* output is included in the .txt file *before* you submit for grading. You will earn a 0 if any or all of it is missing. That includes the carriage control characters.

- Submit the job output in a .txt file on Blackboard as usual.

Additionally, your program must contain the specified external subprograms and `DSECT`'s. You will lose *significant* points if your program does not.

This is a substantial assignment, so be methodical and pace yourself. Waiting until the 11th hour to start is a sure path to late penalties and a nasty ding in your assignment score percentage.