Homework #2 Sparse Matrix

(due: 11:59pm 10/17/2021)

Overview

In this homework, you need to implement the sparse matrix. Matrices will be sparse in the test cases, meaning most elements are just zeros. You can optimize matrix implementation by taking advantage of sparseness. For example, it may make sense to implement the data structure of a matrix with lists instead of two-dimensional arrays. It is also possible to use sparseness to optimize the implementation of the multiplication and addition operations.

The format of `input.txt` and `output.txt` are given as follows.

Input.txt

Each line in input.txt contains a command. There are seven kinds of commands:

```
Matrix matrix
Init matrix row col
Assign matrix row col value
Print matrix
Delete matrix
Mult matrix1 matrix2 matrix3
Add matrix1 matrix2 matrix3
```

Matrix *m* is used to define a new matrix named *m*. You can store the name and the object (or the pointer to the matrix) into a table. Each name of the matrix is unique.

Init *matrix row column* is used to initialize a *row*$\times$*column* matrix. You can initialize your matrices according to your implementation. (e.g., If you use a two-dimensional array as your matrix, you need to set all elements to zero. On the other hand, if you implement your matrix by the linked list data structure, you just need to set the head of the list to NULL.)

Assign *matrix row col value* is used to assign *value* to the element at *row* row and *col* column.

Delete *matrix* is used to delete *matrix*. You need to clean the resource of the matrix. (e.g., deallocate the object)

Mult *matrix1 matrix2 matrix3 is* used to multiply *matrix1* and *matrix2*, then assign the result to *matrix3*.

Add *matrix1 matrix2 matrix3* is used to add *matrix1* and *matrix2*, then assign the result to *matrix3*.

Print *matrix* is used to print all non-zero elements from *matrix* to output.txt. The format of output.txt is described as below.

output.txt

Each line of output.txt contains non-zero elements in the matrix. Each element is wrapped in parentheses. In the parentheses, the first value is the element's row number, the second is the element's column number, and the last is the element's value.

In the output.txt, the elements should be sorted in ascending order by the row numbers. If two elements have the same row number, the one with the smaller column number should appear first. For example, there are three elements, and the order of the list should be

(0, 0, 1) (0, 1, 1) (1, 0, 3)

```
(row1 column1 value1) (row2 column2 value2) … (rowN columnN valueN)
non-zero elements in matrix_2
…
non-zero elements in matrix_n
```

Example1

Input:

```
Matrix a
Init a 3 3
Assign a 0 0 78469
Matrix b
Matrix c
Init b 3 3
Init c 3 3
Assign b 0 1 12345
Add a b c
Delete a
Print c
```

```
Delete b
Delete c
```

$$a = \begin{bmatrix} 78469 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad b = \begin{bmatrix} 0 & 12345 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$c = a + b = \begin{bmatrix} 78469 & 12345 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Result:

```
(0 0 78469) (0 1 12345)
```

Example2

Input:

```
Matrix a
Matrix b
Matrix c
Init a 1 1000000000
Init b 1000000000 1
Init c 1 1
Assign a 0 200000000 1
Assign b 200000000 0 12345
Mult a b c
Print c
Delete a
Delete b
Delete c
```

Result:

```
(0 0 12345)
```

Constraints

1. 0 < the number of rows, the number of columns < 2^{31} (2147483648)
2. -2^{31} $\leq$ the value of element < 2^{31}. The value of element will be integer.
3. the number of matrices < 40

Files

We provide two files, input.txt and result.txt, to help you debug your code. You can compare your output with result.txt. The test case we provided is different from the one on the code sensor. So, you won't get any points if you print the result directly.

Preloaded Input Data

```
struct tTestData {
        int line_num;
        char data[max_data_size][buffer_size];
};
```

line_num: the number of commands
data: each line of command

Submission of Homework

1. Download hw2.zip in the attachment.
2. Implement sparse matrix
3. Upload your code.cpp or code_shm.cpp to Code Sensor
4. Use the “View” and “Scoreboard” function on CODE SENSOR to make sure your submission is successful.