Assignment 1: Matrix Computation

Data Structures (CS-UH 1050) — Spring 2024

1 Code of Conduct

All assignments are graded and we expect you to adhere to the academic integrity standards of NYU Abu Dhabi. To avoid any confusion regarding this, we briefly state what is and isn't allowed when working on an assignment.

Any documents and program code that you submit must be fully written by yourself. You can discuss your work with fellow students, as long as **these discussions are restricted to general solution techniques, without sharing the overall or specific details.** Put differently, these discussions should not be about concrete code you are writing, nor about specific set of steps or results you wish to submit. Discussions with others should not lead to you possessing the complete or partial solution of others in any form (paper or digital), regardless of who made the solution. You are also not allowed to possess solutions by someone from a different year or course, by someone from another university, or code from the Internet, etc. This also implies that **there is never a valid reason to share your code with fellow students, and that there is no valid reason to publish your code online in any form**. Every student is responsible for the work they submit. If there is any doubt during the grading about whether a student created the assignment themselves (e.g., if the solution matches with high similarity score that of others), **the suspected violations will be reported to the academic administration according to the policies of NYU Abu Dhabi** (see https://students.nyuad.nyu.edu/campus-life/community-standards/policies/academic-integrity/) under the integrity review process.

2 Introduction

In this assignment, you will develop a matrix computation system using C++ that will be used to perform addition and multiplication on matrices. A matrix may be sparse, i.e., a large fraction of its elements may be zeros. For such matrices it is often memory efficient to store only the non-zero entries along with their positions in the matrix. If the matrix has M rows, N columns and V non-zero elements, this memory efficient representation typically requires space proportional to M+N+V instead of M*N.

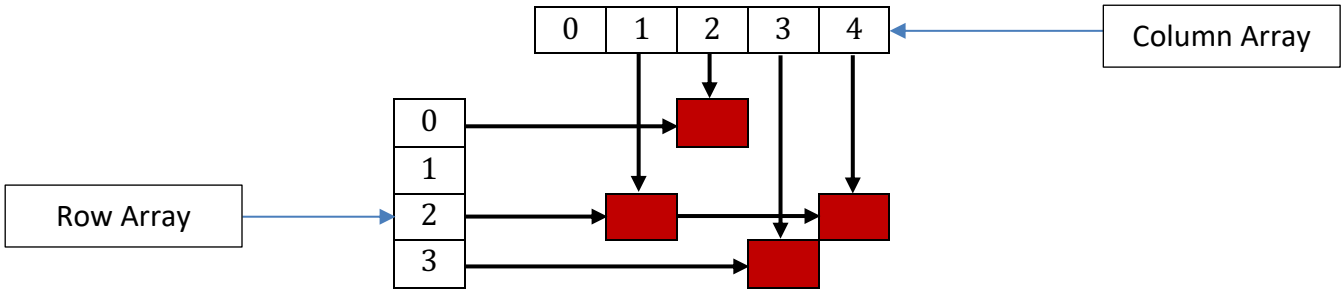
You are supposed to create a program utilizing mainly the following covered topics and concepts:

- File input/output in C++
- Object Oriented Programming
- Linked Lists

Note: You are not allowed to use any built-in STL data structures such as List, Vector, etc.

3 Implementation

A matrix can be stored as Linked Lists as demonstrated in the following example. Each Node of a linked list represents a non-zero entry of the matrix and contains the row index, column index, data value, and pointers to the next non-zero entry to its right (if any) in the same row and the next non-zero entry (if any) below it in the same column For example, the matrix [[0,0,1,0,0], [0,0,0,0,0], [0,1,0,0,1], [0,0,0,1,0]] can be stored as two Arrays of Linked Lists as shown below:



The first non-zero element is at (0, 2). It is pointed to by the header element of row 0. The next non-zero element is at (2, 1), which has a pointer to (2,4). Each non-zero element is pointed to by the header element for the corresponding columns. This way, the actual size of this matrix as built and stored in memory is 4+5+4= 13 instead of 4*5=20!

And as a result, assuming that the matrix entries are integers, a node corresponding to a non-zero entry in the matrix may be defined as follows:

```
Class Node {
   public:
      int row;     // row index
      int column; // column index
      int value;  // value of the non-zero entry
      Node* next; // pointer to the next node within the same row
      Node* down; // pointer to the next node within the same column
}
```

You are provided with a starter code containing class declarations (LinkedMatrix and MatComp classes) and the main function. Your task is to complete the definition and implement all the missing methods of the class. You may add new helper methods in the classes, but you are not allowed to change/remove the definitions of existing methods. The input/output format of the program should be the same as illustrated in the provided screenshots.

Method Description:

When your program starts, the user is presented with the list of possible commands. The user is able to interact with the system using the commands until an exit command is issued.

```
================================================================================
Welcome to the Matrix Computation system!

List of available Commands:
import <file_name>    : Read a matrix from <filename>
export <file_name>    : Write a matrix into <filename>
add                   : Add 2 matrices A and B and store the result in matrix A
multi-by-const <const> : Multiply a matrix by a constant and update the matrix
multi-by-matrix        : Multiply a matrix by another matrix and return the resultant
display               : Display a matrix in the optimized structure
help                  : Display the list of available commands
exit                  : Exit the Program
================================================================================

>_
```

The following features are required to be implemented in the matrix computation system:

void LinkedMatrix::create(int numRows, int numCols)
This method builds an empty matrix of size numRows*numCols based on the optimized matrix data structure.

int LinkedMatrix::getNumRows()
This method returns the number of rows.

int LinkedMatrix::getNumCols()
This method returns the number of columns.

void LinkedMatrix::insertElement(int Row_Indx, int Col_Indx, int value)
This method inserts an element/node with a value, which is not equal to zero, to the matrix at (Row_Indx, Col_Indx). If an element/node already exists at (Row_Indx, Col_Indx), its value is to be updated with the value given as an argument.

void LinkedMatrix::removeElement(int Row_Indx, int Col_Indx)
This method removes the element/node at (Row_Indx, Col_Indx).

void MatComp::importMatrix(LinkedMatrix& newMat, string filename)
This method reads a matrix from the given file into the provided LinkedMatrix object, which follows the optimized representation of a matrix described in the previous section.

```
> import /Desktop/mat1.txt
Done! This is matrix 'A'.
```

void MatComp::multi(LinkedMatrix& Mat, int a)
This method multiplies Matrix “Mat” by the constant ‘a’. The result is stored in “Mat”. In other words, this method updates the matrix provided via the first argument.

```
> multi-by-const 10
Please provide the matrix ID [A,B,C,..]:
A
Done!
```

void MatComp::add(LinkedMatrix& Mat1, LinkedMatrix& Mat2)
This method adds the matrices Mat1 and Mat2, which are of the same size. The result is stored in “Mat1”. In other words, this method updates the matrix provided via the first argument.

```
> add
Please provide the 1st matrix ID [A,B,C,..]:
A
Please provide the 2nd matrix ID [A,B,C,..]:
B
Done!
```

void MatComp::multi(LinkedMatrix& Mat1, LinkedMatrix& Mat2, LinkedMatrix& Mat3)
This method multiplies the matrices Mat1 (of size a*b) and Mat2 (of size b*c) together. That means the number of columns in Mat1 should be equal to the number of rows in Mat2. The resulting matrix is stored in “Mat3”, which would be of size a*c, then displayed along with its assigned ID. If a provided matrix ID does not exist, a message should be displayed saying “No matrix with this ID exists in the system!”.

```
> multi-by-matrix
Please provide the 1st matrix ID [A,B,C,..]:
A
Please provide the 2nd matrix ID [A,B,C,..]:
B

NumRows = 4   NumColumns = 5   NumNonZeroElements = 5
(0,0) = 1
(3,0) = 22
(0,2) = 4
(3,2) = 6
(2,3) = 55
Done! This is matrix 'C'.
```

void MatComp::exportMat(LinkedMatrix& Mat, string filename)
This method writes the matrix "Mat" to a new file.

```
> export /Desktop/mat3.txt
Please provide the matrix ID [A,B,C,..]:
C
Done!
```

void MatComp::display(LinkedMatrix& Mat)
This method displays the provided matrix "Mat". If the provided matrix ID does not exist, a message should be displayed saying "No matrix with this ID exists in the system!".

```
> display
Please provide the matrix ID [A,B,C,..]:
A
NumRows = 4   NumColumns = 5   NumNonZeroElements = 5
(0,0) = 1
(3,0) = 22
(0,2) = 4
(3,2) = 6
(2,3) = 55
Done!
```

Matrix File Format:

A matrix file (.txt), which you can read/import or generate/export, will contain the original dimensions of the matrix (i.e., the number of rows and the number of columns), the number of non-zero elements, and a list of these elements with their addresses. The following is an example of the contents of such a file:

```
Rows=4
Columns=5
Non-ZeroElements=5
(0,0)=1
(3,0)=22
(0,2)=4
(3,2)=6
(2,3)=55
```

Constraints:

- The list of elements in a matrix file must be ordered by column first, and then by row as illustrated in the file example.
- If a non-zero entry becomes zero as a result of an operation, it should be removed from the matrix.
- The matrices should be given IDs based on the order of import and/or generation via multi-by-matrix command. For example, if you import a matrix, import another and then multiply the first one by the second, a third matrix is generated and stored next in the system. Therefore, the ID of the firstly imported matrix is ‘A’, the ID of the secondly imported matrix is ‘B’, and the ID of the generated matrix from multiplication by matrix is ‘C’. If you were to import one more matrix, it would have had ‘D’ as an ID.
- You need to check the validity of ANY input provided by the user and handle errors.
- You need to make sure that the conditions of any operation are met for it to be executed successfully.
- A user should be given up to 3 chances to provide a valid value as an input. In case a user uses the 3 chances without success, then the operation will time-out and the user is back to the main program, where they can enter a command.

4 Grading

Description	**Score (/20)**
Quality in Code Organization & Modularity - Defining ALL the needed classes correctly, with their sets of attributes and methods (including the constructors and destructors) - Creating objects from each class properly - Proper initiation and termination of the system	3
Methods to design and implement with correct execution: import (1.5 pts), export (1.5 pts), add (1.5 pts), multi-by-const (1.5 pts), multi-by-matrix (1.5 pts), display (1.5 pts), create (1 pt), getNumRows (1 pt), getNumCols (1 pt), insertElement (1.5 pts), removeElement (1.5 pts)	15
Proper error handling of missing and invalid input, etc.	1
Properly commenting the code (i.e., code documentation)	1

You should solve and **work individually** on this assignment. The deadline of this assignment is **in 14 days of its release** on NYU Brightspace.

You should compress your **C++ source files** (*.cpp, *.h, makefile) into **a single zip file** before uploading it directly to NYU Brightspace. The zip file should contain:

1. linkedmatrix.h
2. linkedmatrix.cpp
3. matcomp.h
4. matcomp.cpp
5. makefile
6. main.cpp

NO SUBMISSIONS OR RESUBMISSIONS VIA EMAIL WILL BE ACCEPTED. Note that your program should be implemented in C++ and **must be runnable on the Linux, Unix or macOS operating system**. Late submissions will be accepted **only up to 3 days late**, afterwards you will receive zero points. For late submissions, 5% will be deducted from the assignment grade per late day.