Data Structures and Algorithms Lab

Lab 06 **Marks 10**

Instructions

Work on this lab individually. You can use your books, notes, handouts etc. but you are not allowed to borrow anything from your peer student.

Marking Criteria

Show your work to the instructor before leaving the lab to get some or full credit.

What you must do

Program the following task in your C++ compiler and then compile and execute them. *Write **main** function first and keep on testing the functionality of each function once created.*

ADT: Matrix

Write a class (in C++) for 2-dimesional matrices (**Matrix**) of integer type. This class should store the elements of the 2-dimensional matrix in a one-dimensional array of an integer type created dynamically. Thus, you will have to use a mapping mechanism (formula) to store and retrieve the individual elements.

There will be three data members (private) of this class:

- A pointer to integer type (which will be used to allocate memory dynamically)
- An integer to store the number of rows of the matrix
- An integer to store the number of columns of the matrix

Your class should support the following operations:

A. Constructor for creating a **Matrix** of any size (any number of rows and columns). The dimensions (number of rows and columns) will be specified through arguments.

B. Destructor to free any memory resources occupied by an object.

C. `int get(int i, int j)` to get the value of element stored at the *i*th row and *j*th column in the matrix. Also perform bound-checking on the values of *i* and *j*.

D. `void set(int i, int j, int v)` to set the value of the element stored at the *i*th row and *j*th column in the matrix to *v*. Also perform bound-checking on the values of *i* and *j*.

E. `void print(void)` to print the matrix on screen (in 2-D form).

F. `void transpose(void)` to take transpose of the matrix.

G. `void printSubMatrix(int r1, int r2, int c1, int c2)` to display the elements of the sub-matrix specified by the arguments, where *r*1…*r*2 is the range of rows and *c*1…*c*2 is the range of columns to be displayed.

H. `void makeEmpty(int n)` to set the first *n* rows and columns to zero.

I. `void add(Matrix first, Matrix second)` to add two matrices and store the result in the current object (on which this function was called). You may need to change the dimensions of the current object. Keep in mind that two matrices can be added only if the dimensions of both the matrices are same. This function should display an appropriate error message if the two matrices cannot be added.

If you want to add two matrices B and A and store the result in the matrix C, then this function will be called like this: `C.add(A, B)`.

In ***main*** function, create few objects of ***Matrix*** class and demonstrate the working of each function clearly.

☺ ☺ ☺ **BEST OF LUCK** ☺ ☺ ☺