Data Structures & Algorithms LAB – Fall 2015

(BS-SE-F14 Morning & Afternoon)

Lab # 1

Instructions:

- Make sure that there are no ***dangling pointers*** or ***memory leaks*** in your programs.
- Indent your code properly.
- Use meaningful variable and function names. Follow the naming conventions.
- Use meaningful prompt lines/labels for all input/output that is done by your programs.
- Implement all the functions **in the given order**.

Develop a class (in C++) for storing and manipulating 2-dimesional matrices (**`TwoDMatrix`**) of positive integers. There will be three (private) data members of this class:

```
int ** matrix;    // A pointer to the 2-D array of integers (to be allocated dynamically)
int rows;         // Number of rows in the matrix
int columns;      // Number of columns in the matrix
```

You are required to implement the following public member functions of the **`TwoDMatrix`** class:

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

ii. Destructor

iii. **`bool get(int i, int j, int& val)`** 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* i.e. if i or j (or both) are invalid then this function should return false, otherwise this function should store the desired value in the reference parameter val and return true.

iv. **`bool set(int i, int j, int val)`** to set the value of the element stored at the *i*th row and *j*th column in the matrix to *val*. Also perform bound-checking on the values of *i* and *j* i.e. if i or j (or both) are invalid then this function should not update any value and should return false. If both i and j are valid, this function should store val at the specified location and should return true;

v. **`void display()`** to display the matrix on screen.

vi. **`void transpose()`** to take transpose of the matrix. (Hint: Memory deallocation and re-allocation will be required in this function.)

vii. **`void add (const TwoDMatrix& first, const TwoDMatrix& 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 A and B and store the result in the matrix C, then this function will be called like this: **`C.add(A,B)`**.

viii. **`void displaySubMatrix (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. If the range of rows or the range of columns is invalid, this function should display an appropriate error message on screen.

ix. Copy constructor

x. Overloaded assignment operator

xi. **`void multiply (const TwoDMatrix& first, const TwoDMatrix& second)`** to multiply 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.

Write a main function which illustrates the working of **all** the member functions of the **`TwoDMatrix`** class.