

# Data Structures and Algorithms Lab

## Instructions

Work on this lab individually. *Write main function first and keep on testing the functionality of each function once created.*

Program the following tasks in your C++ compiler and then compile and execute them.

Email your solution (.cpp) file only to the following respective recipient till **Friday, March 26, 2021**.

**DO NOT** compress/zip your solution.

The email must be sent from your **official PUCIT email id**, otherwise it will **NOT BE ACCEPTED** and will be marked **ZERO**.

The subject of the email should be the exact name of the lab i.e. **Lab 04. 2 MARKS** will be **DEDUCTED**, otherwise.

Degree	Recipient Email	Subject of Email
BSIT Morning	<a href="mailto:dsaubt01@gmail.com">dsaubt01@gmail.com</a>	Lab 04
BSIT Afternoon	<a href="mailto:dsaubt02@gmail.com">dsaubt02@gmail.com</a>	

You are strictly not allowed to add any other data-member/constructor/function in the class. You are also not allowed to change the name or prototype of any data-member/constructor/function.

## ADT: Matrix

Develop a class (in C++) for 2-dimensional 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 a 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:

- 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.
- Destructor to free any memory resources occupied by an object.
- `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*.
- `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*.
- `void print(void)` to print the matrix on screen (in 2-D form).
- `void transpose(void)` to take transpose of the matrix.
- `void printSubMatrix(int r1, int r2, int c1, int c2)` to display the elements of the sub-matrix specified by the arguments, where *r1*...*r2* is the range of rows and *c1*...*c2* is the range of columns to be displayed.
- `void makeEmpty(int n)` to set the first *n* rows and columns to zero.
- `void subtract(Matrix first, Matrix second)` to subtract 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 subtracted only if the dimensions of both the matrices are same. This function should display an appropriate error message if the two matrices cannot be subtracted.

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