**National University of Computer and Emerging Sciences**



Laboratory Manual

for

Data Structures Lab

|  |  |
| --- | --- |
| Course Instructor | Ms. Syeda Tayyaba Bukhari |
| Lab Instructor(s) | Fariha Maqbool |
| Section | BDS-3A |
| Semester | Fall 2022 |

**Department of Computer Science**

FAST-NU, Lahore, Pakistan

**Objectives:**

In this lab, students will practice:

1. Revision of Templates
2. Revision of Pointers
3. Searching and Sorting

Function templates are special functions that can operate with generic types. This allows us to create a function template whose functionality can be adapted to more than one variable type or class without repeating the code for each type. This is achieved through template parameters. A template parameter is a special kind of parameter that can be used to pass a type as parameter. These function templates can use these parameters as if they were regular types. The format for declaring function templates with type parameters is:

**template <class identifier> function\_declaration;**

While defining a function template the body of the function definition is preceded by a statement **template <class identifier>.** The identifier can then be used as the data type of the parameters, the return type of the function, the data type of local variables and/or the data types of parameters.

**Task 1**

Write Template function for performing arithmetic operation of type int, float, double, long. Main for this function is given below.

**void main( )**

**{**

**int a, b; // this can be float, int or double too**

**char op;**

**cout << "Enter first operand ";**

**cin >> a;**

**cout << "Enter second operand ";**

**cin >> b;**

**cout << "Enter operation ";**

**cin >> op; // op can be +, -, \* or /**

**if (op == '\*' || op == '+' || op == '-' || op == '/')**

**{**

**performOperation(a, b, op);**

**}**

**else**

**{**

**cout << "Wrong operation";**

**}**

**}**

**Task 2:**

Create a class named **Matrix** which has data members (rows, cols) and following functions:

1. **Getters**

Functions to get the values of data members. Make separate getter functions for each data member

1. **Setters**

Functions to set the values of data members. Make separate getter functions for each data member

1. **AllocateMemory**  
   This function takes rows and columns as parameter, create a 2D matrix and return it
2. **Input**
3. This function takes empty matrix as parameter, takes the values from console and save in the matrix
4. **rotateOuterLayerBy1**

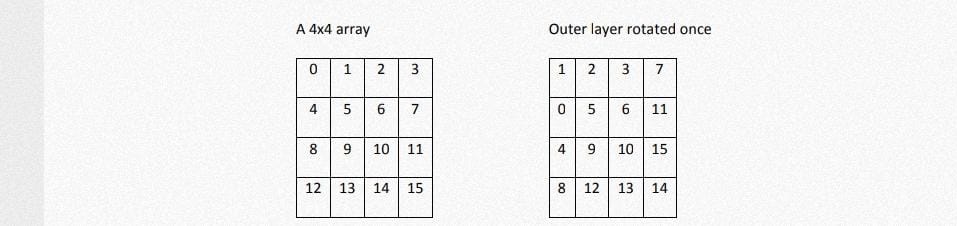
This function takes a 2 dimensional square matrix and its dimension, n, as parameters and rotate the outermost layer by one place, anti-clockwise.

1. **Display**Output the resultant matrix

Create the object of Matrix class in main() function, perform matrix rotation and display the output.

**Constraint**: Your code cannot create an extra array to accomplish the given task. But you can create one or two extra integers.

**Example:**



**Main function**void main()

{

**//take input from user for rows and cols**

**//Create object of Matrix class**

**//set rows and cols values**

**//Call AllocateMemory function**

**//Call Input function  
//Call rotateOuterLayerBy1 function**

**//Call Display function**

**//deallocate matrix**

}