National University of Computer and Emerging Sciences

Laboratory Manual
for
Data Structures Lab

Course Instructor	Mr. Muhammad Naveed
Lab Instructor Lab Demonstrator	Mr. Durraiz Waseem Ms. Adeela Nasir
Section Date	BDS-3A
Semester	Aug 26, 2025
	Fall 2025

Department of Computer Science
FAST-NU, Lahore, Pakistan

Objectives:

In this lab, students will practice:

1. Pointers
2. Templates
3. Time Complexity

Question 1

a. Create a template class "Matrix" with the following members: T** matrix; int rows, columns

You need to define the following member functions:

1. An overloaded constructor which takes the values of rows and columns, and declares the required memory for the matrix. **Matrix(int rows, int columns)**

2. Copy Constructor to deep copy another matrix **Matrix(Matrix const &obj)**

.　　　　　　　　　　　　　　　　　　　　　　　　　　　　　void

3. Insert function to insert an element in the given row number and column number `insertElement(T const& element, int rowNo, int colNo)`

4. An overloaded + operator to add corresponding elements of two matrices. If there is a mismatch of number of rows or columns for the matrices, the operator will print an error. `Matrix<T> operator+(Matrix const& obj)`

5. A function named "print" to print the matrix in a neat and readable way. **void print()**;

6. Transpose function to take transpose of the matrix. (Convert rows into columns and vice versa).

   `void transpose()`

7. A destructor to delete the memory. **~Matrix()**

b. Now test your code for the following objects in your main function:

```
      Matrix<int> m1(2, 3);
m1.insertElement(1,  0,   0);
m1.insertElement(1,  0,   1);
m1.insertElement(1,  0,   2);
m1.insertElement(0,  1,   0);
m1.insertElement(0,  1,   1);

m1.insertElement(0, 1, 2);
m1.transpose();
Matrix<int> m2(2, 3);
m2.insertElement(-1,
0, 0);          m2.insertElement(-1,
0, 1);          m2.insertElement(-1,
0, 2);          m2.insertElement(10,
1, 0);          m2.insertElement(5,
1, 1);          m2.insertElement(1,
```

```
 1, 2);

m2.transpose();
Matrix<int> m3(m2);
Matrix<int> m4(m1 + m3);
m4.transpose();
 m4.print();
```

Question 2

a) You have to design a C++ **template** function **sort**, which takes a dynamic one-dimensional array, its size **n** and an integer **type** which specifies which sorting algorithm to use**.** You need to **implement** functions for following sorting algorithms and also write their **time complexity** of each sorting algorithm.

a. Bubble sort,

b. Insertion sort

c. Selection sort.

b) Function prototype

```
template <typename T>
void sort (T *array, int size, int type)
```

c) Test your function against inputs of different types

a. int

b. char

c. float