**National University of Computer and Emerging Sciences**



Laboratory Manual

for

Data Structures Lab

|  |  |
| --- | --- |
| Course Instructor | Samman Ashraf |
| Lab Instructor(s) | Muhammad Ahmed  Hamza Ayub |
| Section | 4N1 – 4N2 |
| Date | 11-Feb-2023 |
| Semester | Spring 2023 |

**Department of Computer Science**

FAST-NU, Lahore, Pakistan

**Objectives:**

In this lab, students will practice:

1. Stack Implementation using arrays
2. The applications of Stack

**Stack Data Structure**

In this tutorial, you will learn about the stack data structure and its implementation in Python, Java and C/C++.

A stack is a linear data structure that follows the principle of Last In First Out (LIFO). This means the last element inserted inside the stack is removed first.

You can think of the stack data structure as the pile of plates on top of another.

Basic Operations of Stack

There are some basic operations that allow us to perform different actions on a stack.

**Push**: Add an element to the top of a stack

**Pop**: Remove an element from the top of a stack

**IsEmpty**: Check if the stack is empty

**IsFull**: Check if the stack is full

**Peek**: Get the value of the top element without removing it

**Question#1**

1. Implement a template-based stack using Array. The required member methods are:

**bool overflow():** return true if stack is full else false.

**int size()**: returns the count of total element stored in the stack.

**bool isEmpty()**: returns true if the stack is empty else false.

**bool top(T&)**: returns, but does not delete, the topmost element from the stack via the parameter passed by reference. It returns false via a return statement if there is no element in the stack, else it returns true and assigns the top most element to the parameter passed by reference.

**void pop()**: deletes the top most element from the stack. If there is no element, return some error.

**push(T const& e)**: pushes the element “e” on top of the stack.

**Question#2**

**The applications of Stack**

1. Given an expression containing opening and closing braces, brackets, and parentheses; implement a function “isBalanced” to check whether the given expression is a balanced expression or not, using your stack implementation. For example, {[{}{}]}[()], {{}{}}, and []{}() are balanced expressions, but {()}[) and {(}) are not balanced. In your main function test your function using the given examples. bool isBalanced(string exp)
2. Implement a function reverse which reverses the given string using your stack data structure. string reverse(string const);