**National University of Computer and Emerging Sciences**



Laboratory Manual 05

for

Data Structures Lab

|  |  |
| --- | --- |
| Course Instructor | Dr. Zareen Alamgir |
| Lab Instructor(s) | Ms. Fariha Maqbool  Ms. Humna Shabir |
| Section | BCS-3F |
| Semester | Fall 2022 |

**Department of Computer Science**

FAST-NU, Lahore, Pakistan

**Objectives:**

In this lab, students will practice:

1. Queues using Arrays and Linked Lists
2. Stacks using Arrays

# Stacks

**Task 1:**

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

**bool isFull():** 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.

**Task 2:**

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.

bool isBalanced(string exp)

For example, {[{}{}]}[()], {{}{}}, and []{}() are balanced expressions, but {()}[) and {(}) are not balanced. In your main function test your function using the given examples.

# Queues

**Task 1:**

Implement a template-based queue using **Linked List**. The required member methods are:

**bool isFull():** return true if queue 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.

**int front()**: returns the element on Front of queue

**int rear()**: return the element on Rear of queue

**void enqueue()**: Adds an element to queue

**void dequeue()**: Removes an element from queue

**Task 2:**

Given a Queue of integers (Based on **Linked List**). The task is to check if the elements in the queue are consecutive triples after every ‘k’ element. The queue will not be change after calling this function

**Example :**

**Input : 1 2 3 5 7 11 12 13 k = 2**

**Output:** Yes

**Input: 1 2 3 4 5 12 14 15 k = 2**

**Output:** No

**Task 3:**

Given an **Array Queue** of integers of even length. Your task is to arrange the second half elements of queue to the first half in alternative manner.

You can use only one extra queue to complete this task

**Example:**

**Input: 3 2 5 6 7 9 8 4**

**Output: 3 7 2 9 5 8 6 4**