University of Computer and Emerging Sciences

Laboratory Manual

for

Data Structures Lab

Course Instructor	Ma'am Arooj Khalil
Lab Instructors	Mr. Sohaib Ahmad —
Section	BSE-3A
Semester	Fall 2022

Department of Computer Science

FAST-NU, Lahore, Pakistan

NOTE

Zero Tolerance for Plagiarism. Penalty will be given in accordance with the severity of plagiarism. This also includes forwarding the case to the DC Committee.

Objective of this lab:

Linked List

Instructions:

- Make a separate project for each task.
- Indent your code properly.
- Use meaningful variable and function names. Follow the naming conventions.
- Use meaningful prompt lines and labels for all input/output.
- Make sure that there are NO dangling pointers or memory leaks in your program.

Task 1:

Using LinkedList class, implement a template class Stack with the following functionalities

1. Stack() — default constructor.
2. Void Push (T val)
3. bool Pop (T & newval) — if pop is successful, returns true with the popped value stored in newval.
4. bool IsEmpty()
5. bool Top(T & val)

Write a main function to test all the functionalities of the Stack class

Task 2:

Using LinkedList class, implement a template class Queue with the following functionalities

1. Queue() — default constructor.
2. Void Enqueue(T val)
3. bool Dequeue(T & newval) — if pop is successful, returns true with the popped value stored in newval.
4. bool IsEmpty()
5. bool Top(T & val)

Write a main function to test all the functionalities of the Queue class

Task 3:

Infix To Postfix Conversion Using Stack

Write a function that takes an infix expression and returns a Postfix Expression.

string convert(string infix);

Input: ((a+(b*c))-d)

Output: abc*+d-

Input: (a+b) *c

Output: ab+c*

Task 4:

Implement a function **check (char expression[])** using Stack, that determines if a given expression is correctly parenthesized. The function takes in a character array. The function returns true if the brackets are applied correctly and properly balanced and false otherwise. The expression contains: 3 type of brackets ( ) , [] and {}, English alphabets, numbers and operators.

For example,

(x + y) * (w / z)	TRUE
A * { B / C } – () (0)	TRUE
x + y) + (a – c)	FALSE
) a + b * 3 (	FALSE
((*+-))	TRUE
([*+-])	TRUE
({ *a+- })	TRUE
(())(	FALSE