**National University of Computer and Emerging Sciences**



Laboratory Manual

for

Data Structures Lab

|  |  |
| --- | --- |
| Lab Instructor(s) | Anosha Khan |
| Section | BCS-3G |
| Semester | Fall 2024 |

**Department of Computer Science**

FAST-NU, Lahore, Pakistan

**Objectives:**

In this lab, students will practice:

1. Stack

**Question 1**

Write a class Stack that implements a stack data structure using **SLL**. The required member methods are:

1. **Push(const T& item):** Insert a new item onto top of the stack
2. **T Pop():**Removes and returns the top item from the stack.
3. **T TopItem() const:** Returns the top item without removing it.
4. **bool IsEmpty() const:** Returns true if the stack has no elements, otherwise returns false
5. **void Clear():** Removes all elements from the stack, making it empty.
6. **int size():** returns the count of total elements stored in the stack.

**Question 2**

Using the Stack class from Question 1, write a function *EvaluateInfix(string expression)* that takes a mathematical expression in infix notation (e.g., a+(b+d)\*s) and evaluates it, assuming that:

* The expression may contain variables (e.g., a, b, etc.) whose values are provided by the user.
* The operators can include +, -, \*, and /.
* The expression may include parentheses ( and ) to denote precedence.
* Implement the infix evaluation by using **two stacks**: one for operands (numbers or variables) and one for operators.
* Make sure to handle operator precedence and associativity correctly.

**Example:**

For the expression 2 + 3 \* (4 - 1), the function should evaluate and return 11.

**Question3**

Using the Stack class from Question 1, write a function *IsFullyParenthesized(string expression)* that checks whether an input string has balanced and correctly ordered parentheses, brackets, and braces. The function should:

* Support multiple types of brackets: round (), square [], and curly {}.
* Ensure that each opening bracket has a corresponding and correctly positioned closing bracket.
* Use stack operations (Push, Pop) to verify matching brackets.

**Example:**

* For the input string {[()()]}, the function should return True.
* For the input string {[(])}, the function should return False.