Data Structures Lab Manual

Lab 6: Stack

Prepared for: Students of Data Structures

Department of Computer Science  
Fast School of Computing

# Objectives

1. Understand the concept of the stack data structure and its importance in problem-solving.
2. Learn and implement basic stack operations (push, pop, peek, isEmpty, isFull).
3. Explore different implementations of stacks (arrays and linked lists).
4. Apply stacks to solve real-world problems (balancing symbols, expression evaluation, undo/redo, navigation).
5. Develop logical thinking by working on progressively challenging stack problems.

# Lab Outcomes

After successfully completing this lab, students will be able to:  
1. Explain the LIFO (Last In First Out) principle of stacks with real-world analogies.  
2. Implement stacks using arrays and linked lists.  
3. Demonstrate the use of stack operations in coding tasks.  
4. Apply stacks in solving practical problems like:  
 - Balanced parentheses check  
 - Expression evaluation (postfix/infix)  
 - Undo/redo simulation  
 - Browser back/forward navigation  
5. Design and implement a Special Stack that retrieves the minimum element in O(1) time.

**Note: Part 1 and Part 5 will be done on word and remaining will be the coding parts.**

# Lab Task

# Part 1:

• Explain the concept of a stack (Last-In-First-Out).  
• Real-life examples: Stack of plates, undo/redo in editors, browser backtracking.  
• Discuss Operations: push, pop, peek/top, isEmpty, isFull.

# Part 2:

Task 1: Implement stack using arrays with operations push, pop, peek, isEmpty, isFull.  
Test with input: Push 10, 20, 30; Pop 30; Peek 20.

Task 2: Implement stack using a linked list.

# Part 3:

Task 3: Check whether a string of parentheses (), {}, [] is balanced.  
Example: { [ ( ) ] } → Balanced; { ( [ ] ) → Not Balanced.

Task 4: Evaluate a postfix expression using a stack.  
Example: Input: 23\*54\*+9- → Output: 17.

Task 5: Convert an infix expression to postfix using a stack.  
Example: Input: (A+B)\*C → Output: AB+C\*.

# Part 4:

Task 6: (Undo/Redo Simulation): Simulate an undo/redo feature using two stacks.

Task 7: (Browser Back/Forward Simulation): Simulate browser back/forward using two stacks.

# Part 5:

• Discuss real-world uses of stacks (recursion, compiler parsing, memory management).

# Part 6:

Problem: Implement a Special Stack that supports the following operations in O(1) time:  
1. push(x) – Push an element into the stack.  
2. pop() – Remove the top element.  
3. top() – Return the top element.  
4. getMin() – Return the minimum element in the stack.  
  
Constraints:  
- Cannot use extra arrays/lists to scan for minimum each time.  
- Use only one auxiliary stack to keep track of the minimum.  
  
Example Run:  
Push(5), Push(2), Push(8), Push(1)  
getMin() → 1  
pop() → removes 1  
getMin() → 2

# Submission Guidelines

- Submit your .cpp file with proper comments.  
- Make sure your program compiles and runs successfully.