Data Structure and Algorithm Lab Manual (Week-05)

Lab Instructor: Ms. Rabeeya Saleem

Session: 2024 (Fall 2025)

Implement LinkedList class in C++ which must have following functions.

Stack Structure

```
#include <iostream>
using namespace std;

#define SIZE 10   // Maximum size of stack to 10

class Stack {
private:
   int arr[SIZE];  // Array to store stack elements
   int top;       // Index of the top element

public:
   // Constructor
   Stack() {
      top = -1;   // Stack is initially empty
   }

};
```

Push an Element in Stack

```
   // Push: Add an element to the stack (Add this function within stack class)
   void push(int value) {
      if (top == SIZE - 1) {
         cout << "Stack Overflow! Cannot push " << value << endl;
      } else {
         top++;            // Move top pointer upward
         arr[top] = value;   // Insert value
         cout << value << " pushed into stack." << endl;
      }
   }
```

Pop an Element in Stack

```
   // Pop: Remove the top element
   void pop() {
      if (top == -1) {
         cout << "Stack Underflow! Nothing to pop." << endl;
      } else {
         cout << arr[top] << " popped from stack." << endl;
         top--;   // Move top pointer downward
      }
   }
```

Peek Element of Stack

```
// Peek: Show the top element without removing it
void peek() {
    if (top == -1) {
        cout << "Stack is empty!" << endl;
    } else {
        cout << "Top element is: " << arr[top] << endl;
    }
}
```

Display Elements of Stack

```
// Display: Print all elements in the stack
void display() {
    if (top == -1) {
        cout << "Stack is empty!" << endl;
    } else {
        cout << "Stack elements (top to bottom): ";
        for (int i = top; i >= 0; i--) {
            cout << arr[i] << " ";
        }
        cout << endl;
    }
}
```

Basic Functions of Stack

```
    // isEmpty: Check if stack is empty
bool isEmpty() {
    return (top == -1);
}

// Return the number of elements in the stack
int size() {
    return top + 1;
}
```

Basic Functions of Stack

```
void reverseDisplay() {
    if (isEmpty()) {
        cout << "Stack is empty!" << endl;
    } else {
        cout << "Stack elements (bottom to top): ";
        for (int i = 0; i <= top; i++) {
            cout << arr[i] << " ";
        }
        cout << endl;
    }
}
```

Main Driver Function for Stack Class

```
int main() {
    Stack s;

    s.push(10);
    s.push(20);
    s.push(30);
    s.push(40);
    s.push(50);
    s.push(60); // will overflow

    cout << endl;
    s.display();
    s.peek();
    s.getBottom();

    cout << "Current size: " << s.size() << endl;

    cout << endl;
    s.pop();
    s.display();

    cout << endl;
    s.reverseDisplay();

    cout << endl;
    s.clear();
    s.display();

    return 0;
}
```

Problems1-7:

Write an algorithm to reverse words in the sentence. Note: Do not tokenize the word, rather traverse the sentence character by character, Use the stack to solve the problem.	**Input:** I am from University of Engineering and Technology Lahore **Output:** Lahore Technology and Engineering of University from am I
Write a program to check if a given expression containing `()`, `[]`, and `{}` is balanced. Use a stack to solve this problem.	Input: {[()]} Output: Balanced Input: {[(])} Output: Not Balanced
Write a C++ program that removes **adjacent duplicate characters** from a given string using a **stack**.	**Input: "aaabccddd"** **Output: abcd**

Write a C++ program to **implement the `GetMiddle()`** — returns the **middle element** of the stack without removing it. Notes: • you must use only stack operations (like push, pop, peek) and recursion — **no direct indexing** or loops on an array. • If the stack has an even number of elements, consider the **second middle** element as the "middle".	`Input: [50, 40, 30, 20, 10]` Output: Middle = `30` After Delete → `[50, 40, 20, 10]`
Write a C++ program to **implement the `DeleteMiddle()`** — removes the **middle element** from the stack Notes: • you must use only stack operations (like push, pop, peek) and recursion — **no direct indexing** or loops on an array. • If the stack has an even number of elements, consider the **second middle** element as the "middle".	`Input: [50, 40, 30, 20, 10]` Output: After Delete → `[50, 40, 20, 10]`

Write a C++ program to **convert an infix expression** (which may contain operands, operators `+`, `-`, `*`, `/`, and parentheses `()`) into its equivalent **postfix (Reverse Polish) expression** using a **stack**.	Input: A+B*C
	Output: ABC*+
	Input:
	`A+(B*C-(D/E))`
	`Output:`
	`A B C * D E / - +`

Write a **C++ program** to evaluate a **postfix expression** (also known as **Reverse Polish Notation**) using a **stack**.	Input: Postfix: `5 3 + 8 2 - *` Output: Evaluation: `(5 + 3) * (8 - 2)` → `8 * 6` = **48**