LAB SESSION 04

Stack Implementation Using Arrays

THEORY

A stack is a linear data structure that follows the Last In First Out (LIFO) principle. The element that is inserted last is the one that gets removed first. Imagine a pile of books: you place one book on top of the other, and you always remove the top one first — this is how a stack operates.
There are two primary operations associated with stacks:

- push() – to insert an element
- pop() – to remove the top element

Other useful operations:

- peek() or top() – returns the top element without removing it
- isEmpty() – checks if the stack is empty
- isFull() – checks if the stack is full (in case of array implementation)

Stack Using Arrays
In an array-based stack, a fixed-size array is used to store the stack elements. A variable top is used to track the index of the last inserted element.
Example:

```
int stack[SIZE];
int top = -1; // Empty stack
```

Each push() increases top, while each pop() decreases it.

Stack Overflow and Underflow

- **Overflow** occurs when trying to push into a full stack.
- **Underflow** occurs when trying to pop from an empty stack.

Applications of Stack

- Undo feature in editors
- Reversing strings
- Balancing symbols (brackets, parentheses)
- Function call management (call stack)
- Expression evaluation (postfix, prefix)

PROCEDURE

1. Define a fixed-size array and initialize top = -1.
2. Implement push, pop, and display operations.
3. Add boundary checks for overflow and underflow.
4. Write a menu-driven program for stack operations.
5. Compile and test with sample inputs.

Algorithm for PUSH(item)

```
1. Check if top == SIZE - 1.
       If true → Print "Stack Overflow" and stop.

2. Otherwise:
       o Increment top ← top + 1.
       o Set stack[top] ← item.
       o Print "Item inserted successfully".
```

Algorithm for POP()

```
1. Check if top == -1.
       o If true → Print "Stack Underflow" and stop.

2. Otherwise:
       o Print "Deleted element: stack[top]".
       o Decrement top ← top - 1.
```

Algorithm for DISPLAY()

```
1. Check if top == -1.

       o If true → Print "Stack is empty" and stop.

2. Otherwise:
       o Print "Stack elements are:".
       o For i from top down to 0:
             ▪ Print stack[i].
```

EXERCISE

1. Modify the code to take user input for all operations.
2. Implement peek() operation.
3. Convert an infix expression to postfix using a stack.
4. Check for balanced parentheses in an expression.
5. Display stack elements without modifying them.