**Assigment- 1 Date: 30/01/2024**

**Pratical-1:**

**Aim:** Implement a stack using any programming language of your choice. Perform basic operations such as push, pop, and peek. Demonstrate the stack’s behavior with a sequence of push and pop operations.

**Push Operation:**

The push operation adds an element to the top of the stack. In C, we can implement a stack using an array or a linked list. Here's an example of the push operation using an array-based implementation:

**Code:**

**#define MAX\_SIZE 100**

**int stack[MAX\_SIZE];**

**int top = -1;**

**void push(int element) {**

**if (top == MAX\_SIZE - 1) {**

**printf("Stack Overflow: Cannot push element, stack is full.\n");**

**return;**

**}**

**stack[++top] = element;**

**}**

**Pop Operation:**

The pop operation removes the top element from the stack. Here's an example of the pop operation using the same array-based implementation:

**Code:**

**int pop() {**

**if (top == -1) {**

**printf("Stack Underflow: Cannot pop element, stack is empty.\n");**

**return -1;**

**}**

**return stack[top--];**

**}**

**Peek Operation:**

The peek operation retrieves the top element from the stack without removing it. Here's an example of the peek operation using the array-based implementation:

**Code:**

**int peek() {**

**if (top == -1) {**

**printf("Stack is empty.\n");**

**return -1;**

**}**

**return stack[top];**

**}**

**All Operation Code:**

**#include <stdio.h>**

**int main() {**

**push(10);**

**push(20);**

**push(30);**

**printf("Peek: %d\n", peek());**

**printf("Pop: %d\n", pop());**

**printf("Pop: %d\n", pop());**

**push(40);**

**printf("Peek: %d\n", peek());**

**printf("Pop: %d\n", pop());**

**printf("Pop: %d\n", pop());**

**return 0;**

**}**

**Output:**

