ASSISMENT 3

Student Name: Bhaskar Kumar **UID: 23BCS14337**
Branch:BE-CSE **Section/Group: KRG-3B**
Semester: 5 **Date of Performance:31-07-25**
Subject Name: DAA
Subject Code:23CSH-301

1. **Aim: WAP of Stack using Array and LinkedList.**

2. **Objective: To develop a program showing insertion and deletion in a Stack (using C++) using array and linked list.**

   ->Insertion(Push) in Stack

   ->Deletion(Pop) in Stack

   Stack follows LIFO(Last In First Out)

   - **Push - Insert at begin**
   - **Pop – Remove from top**

3. **CODE:**

```
#include <bits/stdc++.h>
using namespace std;

//using Array
class StackArray {
   int top;
   int capacity;
   int *arr;

public:

   StackArray(int n) {
      capacity = n;
      arr = new int[n];
      top = -1;
   }
```

```
    void push(int val) {
        if (top == capacity - 1) {
            cout << "Stack Overflow!" << endl;
            return;
        }
        arr[++top] = val;
        cout << val << " pushed into stack\n";
    }

    void pop() {
        if (top == -1) {
            cout << "Stack Underflow!" << endl;
            return;
        }
        cout << arr[top--] << " popped from stack\n";
    }

    void display() {
        if (top == -1) {
            cout << "Stack is empty!\n";
            return;
        }
        cout << "Stack elements: ";
        for (int i = top; i >= 0; i--) {
            cout << arr[i] << " ";
        }
        cout << endl;
    }
};

//Using Linked List
struct Node {

    int data;

    Node* next;

    Node(int val) : data(val), next(NULL) {}

};

class StackLinkedList {

    Node* top;

public:
```

```
StackLinkedList() { top = NULL; }

void push(int val) {
    Node* newNode = new Node(val);
    newNode->next = top;
    top = newNode;
    cout << val << " pushed into stack\n";
}

void pop() {
    if (!top) {
        cout << "Stack Underflow!\n";
        return;
    }
    cout << top->data << " popped from stack\n";
    Node* temp = top;
    top = top->next;
    delete temp;
}

void display() {
    if (!top) {
        cout << "Stack is empty!\n";
        return;
    }
    cout << "Stack elements: ";
```

```
 Node* curr = top;
        while (curr) {
            cout << curr->data << " ";
            curr = curr->next;
        }
        cout << endl;
    }
};
```

OUTPUT:

```
Output
--- Stack Using Array ---
10 pushed into stack
20 pushed into stack
30 pushed into stack
Stack elements: 30 20 10
30 popped from stack
Stack elements: 20 10

--- Stack Using Linked List ---
100 pushed into stack
200 pushed into stack
300 pushed into stack
Stack elements: 300 200 100
300 popped from stack
Stack elements: 200 100
```

4.Learning Outcomes:

- Learned the concept and working of the **Stack data structure (LIFO)**.
- Implemented **stack operations using both array and linked list** in C++.
- Understood **push, pop, peek** operations and **overflow/underflow handling**.
- Analyzed the **time complexity and memory usage** of array vs linked list stack implementations.

5. Algorithm Analysis: