**Name : Tanishq Thuse**

**Branch : SY-CS(AI)**

**Div : B**

**Roll No. : 60**

**Subject : ADS Assignment-5**

**Title : Implementation of Stack using Arrays and LinkedList**

**Q1) Implementation of Stack using arrays :**

**Code :**

#include <stdio.h>

#include <stdbool.h>

#define MAXSIZE 100 *// Define the maximum size of the stack*

*// Structure to represent a stack*

typedef struct {

    int data[MAXSIZE]; *// Array to hold stack elements*

    int top; *// Index of the top element*

} Stack;

*// Function to initialize the stack*

void initialize(Stack \*s) {

    s->top = -1; *// Set top to -1 indicating the stack is empty*

}

*// Function to check if the stack is empty*

bool isEmpty(Stack \*s) {

    return s->top == -1;

}

*// Function to check if the stack is full*

bool isFull(Stack \*s) {

    return s->top == MAXSIZE - 1;

}

*// Function to add an element to the stack (push operation)*

void push(Stack \*s, int element) {

    if (isFull(s)) {

        printf("Stack Overflow\n");

    } else {

        s->top++; *// Increment the top index*

        s->data[s->top] = element; *// Add the new element to the top*

        printf("%d pushed to stack\n", element);

    }

}

*// Function to remove and return the top element from the stack (pop operation)*

int pop(Stack \*s) {

    if (isEmpty(s)) {

        printf("Stack Underflow\n");

        return -1; *// Return -1 to indicate error*

    } else {

        int element = s->data[s->top]; *// Get the top element*

        s->top--; *// Decrement the top index*

        return element;

    }

}

*// Function to return the top element without removing it (peek operation)*

int peek(Stack \*s) {

    if (isEmpty(s)) {

        printf("Stack is empty\n");

        return -1; *// Return -1 to indicate error*

    } else {

        return s->data[s->top]; *// Return the top element*

    }

}

*// Main function to demonstrate stack operations*

int main() {

    Stack s;

    initialize(&s);

    push(&s, 10);

    push(&s, 20);

    push(&s, 30);

    printf("Top element is %d\n", peek(&s));

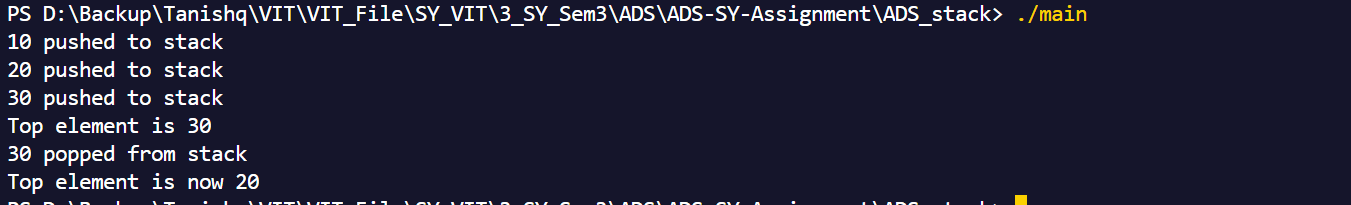
    printf("%d popped from stack\n", pop(&s));

    printf("Top element is now %d\n", peek(&s));

    return 0;

}

**Output :**

****

**Q2) Implementation of Stack using Linked List**

**Code :**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

*// Define a node in the linked list*

typedef struct Node {

    int data; *// Data to store in the node*

    struct Node\* next; *// Pointer to the next node*

} Node;

*// Define a stack structure with a pointer to the top of the stack*

typedef struct {

    Node\* top;

} Stack;

*// Function to initialize the stack*

void initialize(Stack\* s) {

    s->top = NULL; *// Set top to NULL indicating the stack is empty*

}

*// Function to check if the stack is empty*

bool isEmpty(Stack\* s) {

    return s->top == NULL;

}

*// Function to push an element onto the stack*

void push(Stack\* s, int element) {

    Node\* newNode = (Node\*)malloc(sizeof(Node)); *// Allocate memory for a new node*

    if (newNode == NULL) {

        printf("Memory allocation failed\n");

        return;

    }

    newNode->data = element; *// Set the node's data*

    newNode->next = s->top; *// Point the new node's next to the current top*

    s->top = newNode; *// Update the top to be the new node*

    printf("%d pushed to stack\n", element);

}

*// Function to pop an element from the stack*

int pop(Stack\* s) {

    if (isEmpty(s)) {

        printf("Stack Underflow\n");

        return -1; *// Return -1 to indicate error*

    }

    Node\* temp = s->top; *// Temporary pointer to hold the top node*

    int poppedData = temp->data; *// Store the data to return*

    s->top = s->top->next; *// Update top to the next node*

    free(temp); *// Free memory of the old top node*

    return poppedData;

}

*// Function to get the top element of the stack*

int peek(Stack\* s) {

    if (isEmpty(s)) {

        printf("Stack is empty\n");

        return -1; *// Return -1 to indicate error*

    }

    return s->top->data; *// Return the data of the top node*

}

*// Main function to demonstrate stack operations*

int main() {

    Stack s;

    initialize(&s);

    push(&s, 10);

    push(&s, 20);

    push(&s, 30);

    printf("Top element is %d\n", peek(&s));

    printf("%d popped from stack\n", pop(&s));

    printf("Top element is now %d\n", peek(&s));

    return 0;

}

Output :

