Implementation of Stack using Array

and Linked List Implementation

Write a C program to implement a stack using Array and linked List implementation and execute the following operation on stack.

(i) Push an element into a stack

(ii) Pop an element from a stack

(iii) Return the Top most element from a stack

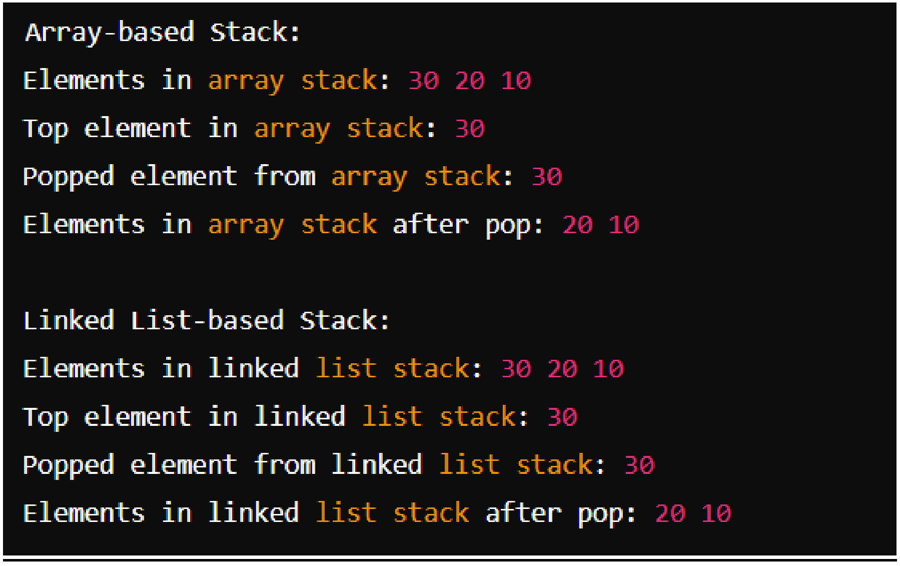
(iv) Display the elements in a stack

PROGRAM:

```
#include <stdio.h>

#include <stdlib.h>


#define MAX 100


// Stack structure using array

struct ArrayStack {

   int arr[MAX];

   int top;

};


// Node structure for the stack using linked list

struct Node {

   int data;

   struct Node* next;

};


// Functions for Array-based Stack


// Initialize the array stack

void initArrayStack(struct ArrayStack* stack) {

   stack->top = -1;

}
```

```
// Check if the array stack is full
int isArrayFull(struct ArrayStack* stack) {
    return stack->top == MAX - 1;
}

// Check if the array stack is empty
int isArrayEmpty(struct ArrayStack* stack) {
    return stack->top == -1;
}

// Push an element onto the array stack
void arrayPush(struct ArrayStack* stack, int data) {
    if (isArrayFull(stack)) {
        printf("Array stack overflow\n");
        return;
    }
    stack->arr[++stack->top] = data;
}

// Pop an element from the array stack
int arrayPop(struct ArrayStack* stack) {
    if (isArrayEmpty(stack)) {
        printf("Array stack underflow\n");
        return -1;
    }
    return stack->arr[stack->top--];
}

// Get the top element of the array stack
int arrayTop(struct ArrayStack* stack) {
    if (isArrayEmpty(stack)) {
        printf("Array stack is empty\n");
        return -1;
    }
    return stack->arr[stack->top];
}

// Display the elements in the array stack
void arrayDisplay(struct ArrayStack* stack) {
    if (isArrayEmpty(stack)) {
        printf("Array stack is empty\n");
        return;
    }
    for (int i = stack->top; i >= 0; i--) {
        printf("%d ", stack->arr[i]);
    }
    printf("\n");
}

// Functions for Linked List-based Stack

// Push an element onto the linked list stack
void listPush(struct Node** top, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (!newNode) {
        printf("Linked list stack overflow\n");
        return;
    }
    newNode->data = data;
    newNode->next = *top;
    *top = newNode;
}

// Pop an element from the linked list stack
int listPop(struct Node** top) {
    if (*top == NULL) {
        printf("Linked list stack underflow\n");
        return -1;
    }
    struct Node* temp = *top;
    *top = (*top)->next;
    int popped = temp->data;
    free(temp);
    return popped;
}

// Get the top element of the linked list stack
int listTop(struct Node* top) {
    if (top == NULL) {
        printf("Linked list stack is empty\n");
        return -1;
    }
    return top->data;
}

// Display the elements in the linked list stack
void listDisplay(struct Node* top) {
    if (top == NULL) {
        printf("Linked list stack is empty\n");
        return;
    }
    struct Node* temp = top;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// Main function
int main() {
    // Array-based stack
    struct ArrayStack arrayStack;
    initArrayStack(&arrayStack);

    // Linked list-based stack
    struct Node* listTopNode = NULL;

    // Operations on array-based stack
    printf("Array-based Stack:\n");
    arrayPush(&arrayStack, 10);
    arrayPush(&arrayStack, 20);
    arrayPush(&arrayStack, 30);
    printf("Elements in array stack: ");
    arrayDisplay(&arrayStack);

    printf("Top element in array stack: %d\n", arrayTop(&arrayStack));
    printf("Popped element from array stack: %d\n", arrayPop(&arrayStack));
    printf("Elements in array stack after pop: ");
    arrayDisplay(&arrayStack);

    // Operations on linked list-based stack
    printf("\nLinked List-based Stack:\n");
    listPush(&listTopNode, 10);
    listPush(&listTopNode, 20);
    listPush(&listTopNode, 30);
    printf("Elements in linked list stack: ");
    listDisplay(listTopNode);

    printf("Top element in linked list stack: %d\n", listTop(listTopNode));
    printf("Popped element from linked list stack: %d\n", listPop(&listTopNode));
    printf("Elements in linked list stack after pop: ");
    listDisplay(listTopNode);

    return 0;
}
```

OUTPUT:



```
Array-based Stack:
Elements in array stack: 30 20 10
Top element in array stack: 30
Popped element from array stack: 30
Elements in array stack after pop: 20 10

Linked List-based Stack:
Elements in linked list stack: 30 20 10
Top element in linked list stack: 30
Popped element from linked list stack: 30
Elements in linked list stack after pop: 20 10
```