NAME:SWETHA.J

ROLL NO:230701357

EX-4: Implementation of Stack using Array and Linked List

```
#include <stdio.h>
#include <stdlib.h>

// Structure for node in linked list implementation
struct Node {
    int data;
    struct Node* next;
};

// Structure for stack using linked list implementation
struct StackLL {
    struct Node* top;
};

// Structure for stack using array implementation
struct StackArray {
    int* array;
    int top;
    int capacity;
};

// Function to initialize stack using linked list implementation
struct StackLL* createStackLL() {
    struct StackLL* stack = (struct StackLL*)malloc(sizeof(struct
StackLL));
    stack->top = NULL;
    return stack;
}

// Function to initialize stack using array implementation
struct StackArray* createStackArray(int capacity) {
    struct StackArray* stack = (struct StackArray*)malloc(sizeof(struct
StackArray));
    stack->capacity = capacity;
    stack->top = -1;
    stack->array = (int*)malloc(stack->capacity * sizeof(int));
    return stack;
}

// Function to check if the stack is empty (linked list implementation)
int isEmptyLL(struct StackLL* stack) {
    return stack->top == NULL;
}

// Function to check if the stack is empty (array implementation)
int isEmptyArray(struct StackArray* stack) {
    return stack->top == -1;
}
```

```
// Function to push element into stack using linked list implementation
void pushLL(struct StackLL* stack, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = stack->top;
    stack->top = newNode;
}

// Function to push element into stack using array implementation
void pushArray(struct StackArray* stack, int data) {
    if (stack->top == stack->capacity - 1) {
        printf("Stack Overflow\n");
        return;
    }
    stack->array[++stack->top] = data;
}

// Function to pop element from stack using linked list implementation
int popLL(struct StackLL* stack) {
    if (isEmptyLL(stack)) {
        printf("Stack Underflow\n");
        return -1;
    }
    struct Node* temp = stack->top;
    int data = temp->data;
    stack->top = stack->top->next;
    free(temp);
    return data;
}

// Function to pop element from stack using array implementation
int popArray(struct StackArray* stack) {
    if (isEmptyArray(stack)) {
        printf("Stack Underflow\n");
        return -1;
    }
    return stack->array[stack->top--];
}

// Function to return top element from stack using linked list
implementation
int peekLL(struct StackLL* stack) {
    if (isEmptyLL(stack)) {
        printf("Stack is empty\n");
        return -1;
    }
    return stack->top->data;
}

// Function to return top element from stack using array implementation
int peekArray(struct StackArray* stack) {
    if (isEmptyArray(stack)) {
        printf("Stack is empty\n");
        return -1;
    }
    return stack->array[stack->top];
}
```

```
// Function to display elements in stack using linked list
implementation
void displayLL(struct StackLL* stack) {
    if (isEmptyLL(stack)) {
        printf("Stack is empty\n");
        return;
    }
    struct Node* temp = stack->top;
    printf("Elements in stack: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// Function to display elements in stack using array implementation
void displayArray(struct StackArray* stack) {
    if (isEmptyArray(stack)) {
        printf("Stack is empty\n");
        return;
    }
    printf("Elements in stack: ");
    for (int i = stack->top; i >= 0; i--) {
        printf("%d ", stack->array[i]);
    }
    printf("\n");
}

int main() {
    // Test linked list implementation
    struct StackLL* stackLL = createStackLL();
    pushLL(stackLL, 1);
    pushLL(stackLL, 2);
    pushLL(stackLL, 3);
    displayLL(stackLL);
    printf("Top element: %d\n", peekLL(stackLL));
    printf("Popped element: %d\n", popLL(stackLL));
    displayLL(stackLL);

    // Test array implementation
    struct StackArray* stackArray = createStackArray(5);
    pushArray(stackArray, 4);
    pushArray(stackArray, 5);
    pushArray(stackArray, 6);
    displayArray(stackArray);
    printf("Top element: %d\n", peekArray(stackArray));
    printf("Popped element: %d\n", popArray(stackArray));
    displayArray(stackArray);

    return 0;
}
```

Output

1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT

Enter your choice : 1

Enter the element : 10

1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT

Enter your choice : 1

Enter the element : 20

1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT

Enter your choice : 1

Enter the element : 30

1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT

Enter your choice : 1

Enter the element : 40

1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT

Enter your choice : 1

Enter the element : 50

1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT

Enter your choice : 1

Enter the element : 60

Stack Overflow...!

1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT

Enter your choice : 4

50 40 30 20 10

1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT

Enter your choice : 3

50

1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT

Enter your choice : 2

50

1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT

Enter your choice : 2

40

1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT

Enter your choice : 2

30

1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT

Enter your choice : 2

20

1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT

Enter your choice : 2

10

1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT

Enter your choice : 2

Stack Underflow...!

1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT

Enter your choice : 5