Folder SEM 3\Exp5

2 printable files

(file list disabled)

SEM 3\Exp5\Stack_ARR.c

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

#define MAX 100 // Maximum size of the stack

// Stack structure using arrays
struct StackArray
{
    int top;
    int arr[MAX];
};

// Function to create a stack
struct StackArray *createStack()
{
    struct StackArray *stack = (struct StackArray *)malloc(sizeof(struct StackArray));
    stack->top = -1; // Initialize the top index
    return stack;
}

// Check if the stack is full
int isFull(struct StackArray *stack)
{
    return stack->top == MAX - 1;
}

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

// Push an element onto the stack
void push(struct StackArray *stack, int value)
{
    if (isFull(stack))
    {
        printf("Stack overflow!\n");
        return;
    }
    stack->arr[++stack->top] = value;
    printf("%d pushed onto stack\n", value);
}

```

```c
// Pop an element from the stack
int pop(struct StackArray *stack)
{
    if (isEmpty(stack))
    {
        printf("Stack underflow!\n");
        return -1; // Return -1 for underflow
    }
    return stack->arr[stack->top--];
}

// Peek at the top element of the stack
int peek(struct StackArray *stack)
{
    if (isEmpty(stack))
    {
        printf("Stack is empty!\n");
        return -1; // Return -1 if empty
    }
    return stack->arr[stack->top];
}

// Display the stack
void display(struct StackArray *stack)
{
    if (isEmpty(stack))
    {
        printf("Stack is empty!\n");
        return;
    }
    printf("Stack elements: ");
    for (int i = stack->top; i >= 0; i--)
    {
        printf("%d ", stack->arr[i]);
    }
    printf("\n");
}

int main()
{
    struct StackArray *stack = createStack();
    int choice, value;

    while (1)
    {
        printf("\nStack Operations (Array Implementation):\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Peek\n");
        printf("4. Display\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            printf("Enter the value to push: ");
            scanf("%d", &value);
            push(stack, value);
            break;
        case 2:
            value = pop(stack);
            if (value != -1)
                printf("Popped value: %d\n", value);
            break;
        case 3:
            value = peek(stack);
            if (value != -1)
                printf("Top value: %d\n", value);
            break;
        case 4:
            display(stack);
            break;
        case 5:
            free(stack);
            exit(0);
        default:
            printf("Invalid choice!\n");
        }
    }

    return 0;
}

```

SEM 3\Exp5\Stack_LL.c

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

// Node structure for the linked list
struct Node
{
    int data;
    struct Node *next;
};

// Stack structure using linked lists
struct StackLinkedList
{
    struct Node *top;
};

// Function to create a stack
struct StackLinkedList *createStack()
{
    struct StackLinkedList *stack = (struct StackLinkedList *)malloc(sizeof(struct StackLinkedList));
    stack->top = NULL; // Initialize the top pointer
    return stack;
}

// Check if the stack is empty
int isEmpty(struct StackLinkedList *stack)
{
    return stack->top == NULL;
}

// Push an element onto the stack
void push(struct StackLinkedList *stack, int value)
{
    struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
    new_node->data = value;
    new_node->next = stack->top;
    stack->top = new_node;
    printf("%d pushed onto stack\n", value);
}

// Pop an element from the stack
int pop(struct StackLinkedList *stack)
{
    if (isEmpty(stack))
    {
        printf("Stack underflow!\n");
        return -1; // Return -1 for underflow
    }
    struct Node *temp = stack->top;
    int popped_value = temp->data;
    stack->top = stack->top->next;
    free(temp);
    return popped_value;
}

// Peek at the top element of the stack
int peek(struct StackLinkedList *stack)
{
    if (isEmpty(stack))
    {
        printf("Stack is empty!\n");
        return -1; // Return -1 if empty
    }
    return stack->top->data;
}

// Display the stack
void display(struct StackLinkedList *stack)
{
    if (isEmpty(stack))
    {
        printf("Stack is empty!\n");
        return;
    }
    struct Node *temp = stack->top;
    printf("Stack elements: ");
    while (temp != NULL)
    {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int main()
{
    struct StackLinkedList *stack = createStack();
    int choice, value;

    while (1)
    {
        printf("\nStack Operations (Linked List Implementation):\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Peek\n");
        printf("4. Display\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            printf("Enter the value to push: ");
            scanf("%d", &value);
            push(stack, value);
            break;
        case 2:
            value = pop(stack);
            if (value != -1)
                printf("Popped value: %d\n", value);
            break;
        case 3:
            value = peek(stack);
            if (value != -1)
                printf("Top value: %d\n", value);
            break;
        case 4:
            display(stack);
            break;
        case 5:
            // Free linked list nodes (cleanup)
            while (!isEmpty(stack))
            {
                pop(stack);
            }
            free(stack);
            exit(0);
        default:
            printf("Invalid choice!\n");
        }
    }

    return 0;
}

```