#include <stdio.h>

#include <stdlib.h>

// Node structure for the singly linked list

struct Node {

int data;

struct Node\* next;

};

// Structure for the stack

struct Stack {

struct Node\* top;

};

// Function to create a new node

struct Node\* createNode(int data) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

if (newNode == NULL) {

printf("Memory allocation failed\n");

exit(EXIT\_FAILURE);

}

newNode->data = data;

newNode->next = NULL;

return newNode;

}

// Function to initialize an empty stack

void initializeStack(struct Stack\* stack) {

stack->top = NULL;

}

// Function to check if the stack is empty

int isEmpty(struct Stack\* stack) {

return (stack->top == NULL);

}

// Function to push an element onto the stack

void push(struct Stack\* stack, int data) {

struct Node\* newNode = createNode(data);

newNode->next = stack->top;

stack->top = newNode;

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

}

// Function to pop an element from the stack

int pop(struct Stack\* stack) {

if (isEmpty(stack)) {

printf("Stack is empty\n");

exit(EXIT\_FAILURE);

}

struct Node\* temp = stack->top;

int poppedData = temp->data;

stack->top = temp->next;

free(temp);

return poppedData;

}

// Function to peek at the top element of the stack without removing it

int peek(struct Stack\* stack) {

if (isEmpty(stack)) {

printf("Stack is empty\n");

exit(EXIT\_FAILURE);

}

return stack->top->data;

}

// Function to display the elements of the stack

void displayStack(struct Stack\* stack) {

if (isEmpty(stack)) {

printf("Stack is empty\n");

return;

}

struct Node\* current = stack->top;

printf("Stack elements: ");

while (current != NULL) {

printf("%d ", current->data);

current = current->next;

}

printf("\n");

}

// Main function for testing the stack operations

int main() {

struct Stack stack;

initializeStack(&stack);

push(&stack, 10);

push(&stack, 20);

push(&stack, 30);

displayStack(&stack);

printf("Top element: %d\n", peek(&stack));

printf("Popped element: %d\n", pop(&stack));

displayStack(&stack);

return 0;

}