Stack Implementation Using Arrays

Problem Statement: Implement a stack data structure using an array. Your program should support the following stack operations:

1. Push: Add an element to the top of the stack.
2. Pop: Remove an element from the top of the stack.
3. Peek: Display the top element without removing it.
4. IsEmpty: Check if the stack is empty.
5. IsFull: Check if the stack is full (assume a fixed size).

Assignment Tasks:

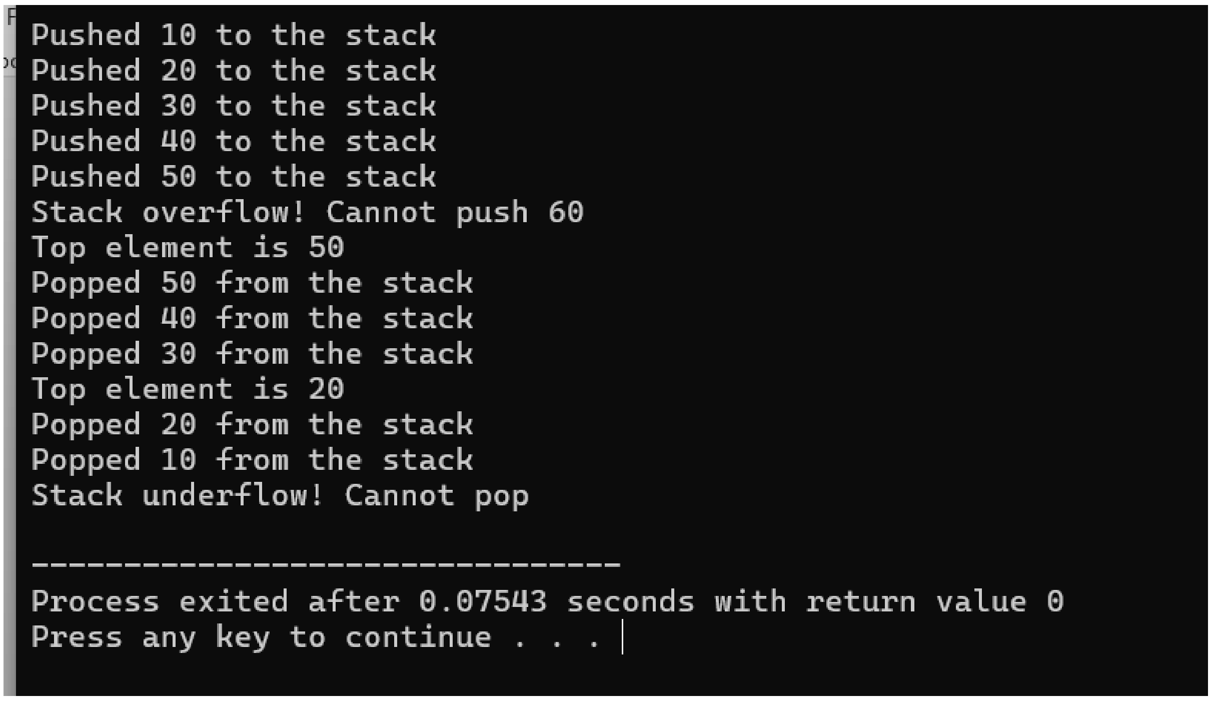
• Write a C program that defines a stack using arrays.
• Implement the stack operations mentioned above.
• Demonstrate stack overflow and underflow conditions.
• Write a main program to test all stack operations.

CODE :

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
struct Stack {
   int items[MAX];
   int top;
};
void initStack(struct Stack *stack) {
   stack->top = -1;
}
int isEmpty(struct Stack *stack) {
   return stack->top == -1;
}
int isFull(struct Stack *stack) {
   return stack->top == MAX - 1;
}
void push(struct Stack *stack, int value) {
   if (isFull(stack)) {
      printf("Stack overflow! Cannot push %d\n", value);
   } else {
      stack->items[++stack->top] = value;
      printf("Pushed %d to the stack\n", value);
   }
}
int pop(struct Stack *stack) {
   if (isEmpty(stack)) {
      printf("Stack underflow! Cannot pop\n");
      return -1;
   } else {
      int value = stack->items[stack->top--];
```

```
        printf("Popped %d from the stack\n", value);
        return value;
    }
}
int peek(struct Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty! Cannot peek\n");
        return -1;
    } else {
        return stack->items[stack->top];
    }
}
int main() {
    struct Stack stack;
    initStack(&stack);
    push(&stack, 10);
    push(&stack, 20);
    push(&stack, 30);
    push(&stack, 40);
    push(&stack, 50);
    push(&stack, 60);
    printf("Top element is %d\n", peek(&stack));
    pop(&stack);
    pop(&stack);
    pop(&stack);
    printf("Top element is %d\n", peek(&stack));
    pop(&stack);
    pop(&stack);
    pop(&stack);
    return 0;
}
```

OUTPUT



```
Pushed 10 to the stack
Pushed 20 to the stack
Pushed 30 to the stack
Pushed 40 to the stack
Pushed 50 to the stack
Stack overflow! Cannot push 60
Top element is 50
Popped 50 from the stack
Popped 40 from the stack
Popped 30 from the stack
Top element is 20
Popped 20 from the stack
Popped 10 from the stack
Stack underflow! Cannot pop

--------------------------------
Process exited after 0.07543 seconds with return value 0
Press any key to continue . . .
```

Stack Implementation Using Linked Lists

Problem Statement: Implement a stack data structure using a linked list. The program should support the following operations:

1. Push: Add an element to the top of the stack.
2. Pop: Remove an element from the top of the stack.
3. Peek: Display the top element without removing it.
4. IsEmpty: Check if the stack is empty.

Assignment Tasks:

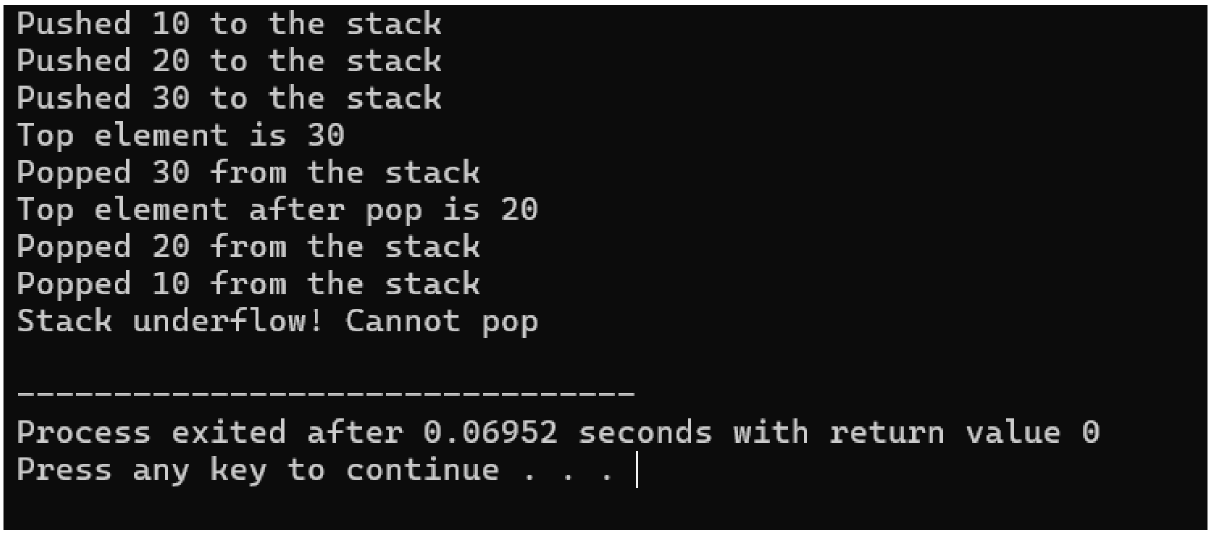
- Write a C program that defines a stack using a singly linked list.
- Implement the stack operations mentioned above.
- Demonstrate stack operations using linked lists.
- Write a main program to test all stack operations.

CODE :

```c
#include <stdio.h>
#include <stdlib.h>
struct Node {
   int data;
   struct Node* next;
};
struct Node* top = NULL;
int isEmpty() {
   return top == NULL;
}
void push(int value) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   if (newNode == NULL) {
      printf("Stack overflow! Cannot push %d\n", value);
      return;
   }
   newNode->data = value;
   newNode->next = top;
   top = newNode;
   printf("Pushed %d to the stack\n", value);
}
int pop() {
   if (isEmpty()) {
      printf("Stack underflow! Cannot pop\n");
      return -1;
   }
   struct Node* temp = top;
   int poppedValue = temp->data;
   top = top->next;
   free(temp);
```

```
    printf("Popped %d from the stack\n", poppedValue);
    return poppedValue;
}
int peek() {
    if (isEmpty()) {
        printf("Stack is empty! Cannot peek\n");
        return -1;
    }
    return top->data;
}
int main() {
    push(10);
    push(20);
    push(30);
    printf("Top element is %d\n", peek());
    pop();
    printf("Top element after pop is %d\n", peek());
    pop();
    pop();
    pop();
    return 0;
}
```

OUTPUT



```
Pushed 10 to the stack
Pushed 20 to the stack
Pushed 30 to the stack
Top element is 30
Popped 30 from the stack
Top element after pop is 20
Popped 20 from the stack
Popped 10 from the stack
Stack underflow! Cannot pop

--------------------------------
Process exited after 0.06952 seconds with return value 0
Press any key to continue . . .
```