LAB - 02 :-

Q. write a program for the swapping two progra using points with a function.

```
# include <stdio.h>
void swap (int * a, int * b)
 {
    int  temp = * a ;
     * a  = * b ;
     * b  =  temp ;
 }
int main()
{
 int num1, num2
 printf(" Enter the first number \n" );
  scanf ("%d", &num1);
  printf (" Enter the second number \n" );
  scanf (" %d", &num2);
 printf (" Before swapping : num1 = %d
          num2 = %d \n", num1, num2)

swap (&num1 , &num2 );
printf (" After swapping : num1 = %d, num2 =
          %d \n", num1, num2);

 return (0);
}
```

⇒ output:

```
Enter the first number = 10
enter the second number = 25
before swapping num1 = 10 , num2 = 25
after swapping num1 = 25 , num2 = 10
```

Q. write a program to implement Dynamic Memory Allocation like Malloc, calloc, free and Realloc.

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

  void* myRealloc (size_t size) {
          return malloc (size);
    }
  void * myRealloc (void * ptr, size_t size)
    {
      return realloc (ptr, size);
    }
  void* my calloc (size_t num, size_t size) {
      return calloc (num, size);
    }
  void my free (void * ptr) {
            free (ptr);
    }
   int main () {
         int *arr1, *arr2;
         size_t size;
         printf (" Enter the size of the array :");
         scanf ("%zu", & size);

     arr1 = (int*) my Malloc (size * size of (int));

     if (arr1 == NULL) {
        printf ("Memory allocation failed.\n");
        return 1;
     }
```

```
printf ("Enter elements of the array :\n");
for (size_t i=0; i< size ; i++) {
        printf (" Element %zu: ", i + 1);
        scanf ("%d", &arr1[i]);
  }
 printf ("Elements of the array (malloc): \n");
  for (size_t i=0; i< size ; i++) {
           print ("%d", arr1[i]);
  }
  printf ("\n");

   size *= 2 ;
   arr2 = (int*) myRealloc (arr1, size * size
        (int));

  if (arr 2 == NULL)
   {
   printf ("Memory reallocation failed. \n");
     myfree (arr 2);
    return 1;

 }
~~printf~~
printF ("Enter additional elements of the allo

For (size_t i = size /2 ; i< size ; i++)
        scanF (" %d", & arr2 [i]);
}
printf ("Elements of the array (realloc): \n")
for (size_t i= 0; i< size ; i++)
       {
          printF ("%d", arr2[i]);
       }
 printF ("\n");
```

```
My free (arr 2);
  return 0;
}
```

→ Output:

```
    Enter the size of the array : 5
    Enter elements of the array:
      Element 1: 1
      Element 2: 2
      Element 3: 3
      Element 4: 4
      Element 5: 5
Elements of the array (malloc):
    1 2 3 4 5
Enter aditional elements of the array:
  Element 6 : 11
  Element 7 : 22
  Element 8 : 33
  Element 9 : 44
  Element 10 : 55
Elements of the array (realloc):
    1 2 3 4 5 11 22 33 44 55
```

Q.3- Write a c program for stack implementation which push, pop, display functions to be implemented.

→

```
# include <stdio.h>
# include <stdlib.h>
# define Max_size 10
struct stack {
      int arr [Max_size];
      int top;
};
```

Date
Page

```
void initialize stack (Struct stack * stack) {
            stack -> top = -1;
      }
   int is Empty (struct stack * stack)
     {
      return stack -> top == -1;
     }
   int is full (struct stack * stack) {
        return stack -> top == Max_ size -1
   }
void push (struct stack *stack, int value) {
     if (is Full (stack)) {
          printf (" stack overflow. cannot push el
  %d \n", value);
        return;
   }
   stack -> arr [++ stack -> top] = value;
   print f (" pushed %d onto the stack. \n
               value);
}
void pop (struct stack * stack) {
        if (is Empty (stack)) {
        printf (" stack underflow. cannot pop fro
   an empty stack. \n");
        return;
   }
printf (" popped %d from the stack. \n", stack->
arr [stack -> top --]);
  }
void display (Struct stack * stack)
```

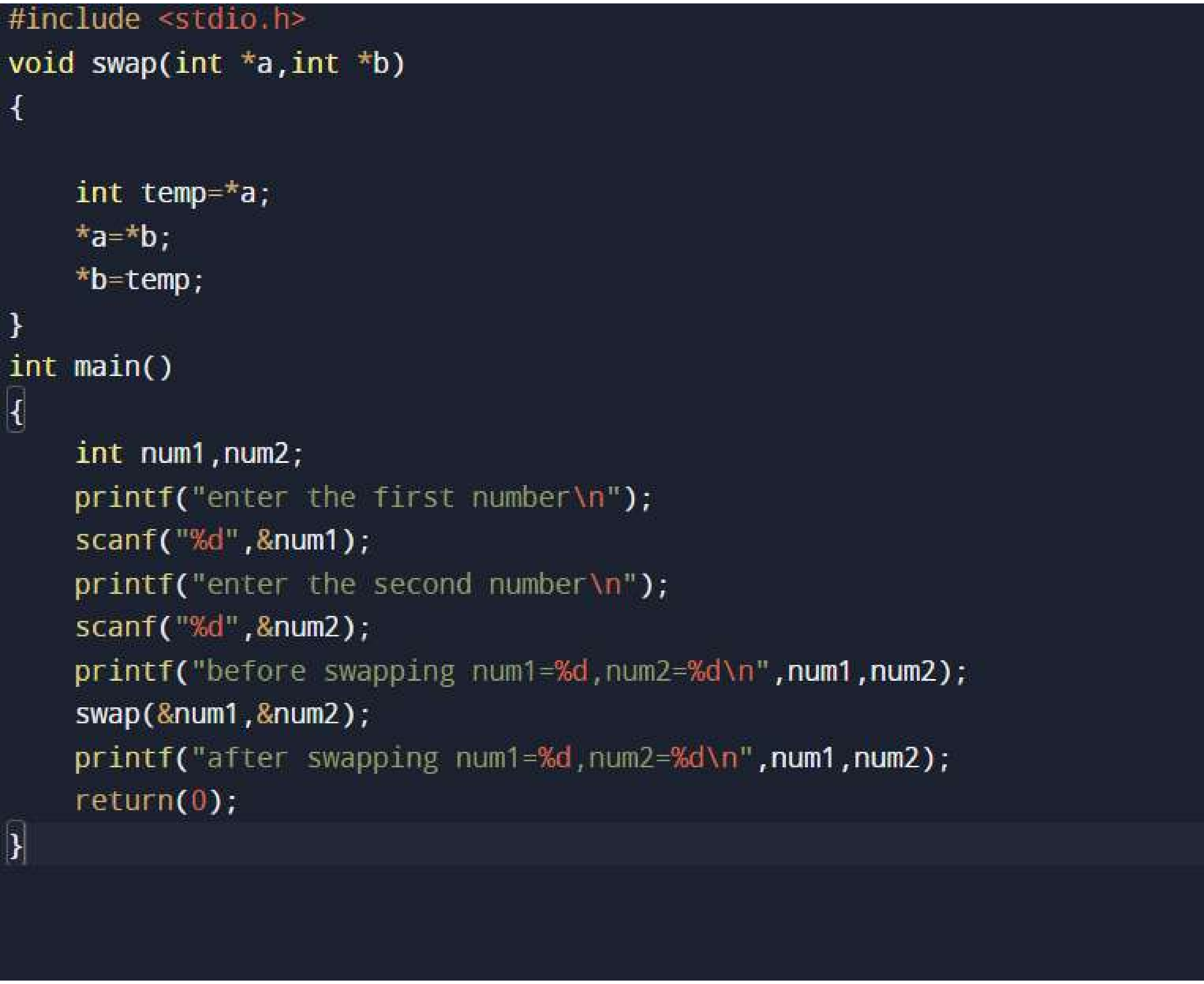
```
{
if (is empty (stack)) {
   printf ("stack is empty. \n");
   return;
}
printf ("stark elements :");
from (int = 0 ; i <= stack → top ; i++)
{
  printf("%d ", stack → arr [i]);
}
 printf ("\n");
}
int  main() {
     struct stack mystack ;
    initialize stack (& mystack);

     push (&my stack , 5) :
     push (&my stack, 10) ;
     push (&my stack, 15);
     display (&my stack);

    pop (&my stack);
    display (& my stack) :

     push (& Mystack , 20) ;
     display (&my stack) ;

     return 0;
}
```

Date __/__/__
Page ____

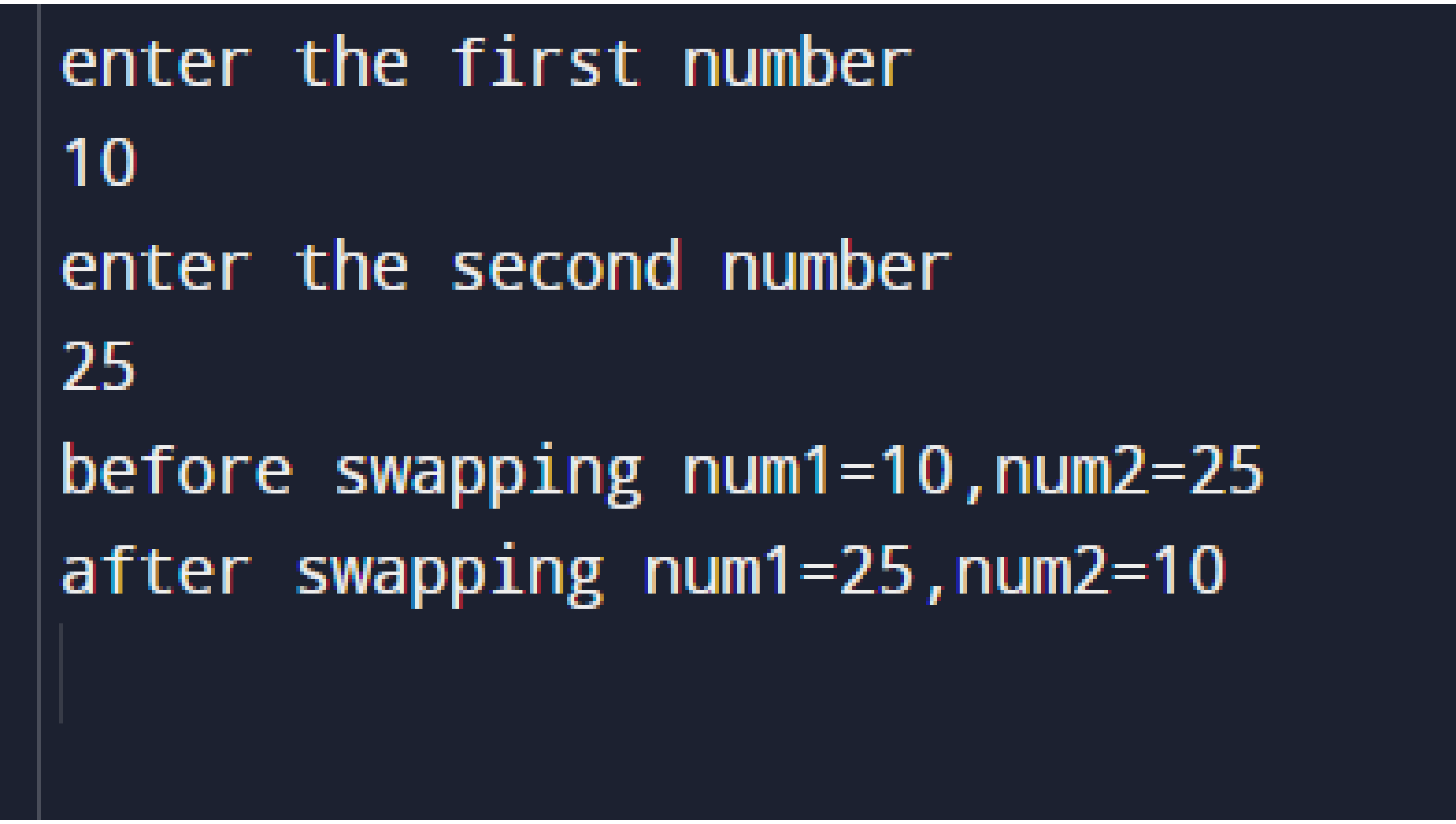
Output :

```
pushed 5 onto the stack
pushed 10 onto the stack
pushed 15 onto the stack
stack elements : 5 10 15
stack elements : 5 10
pushed 20 onto the stack
stack elements : 5 10 20.
```

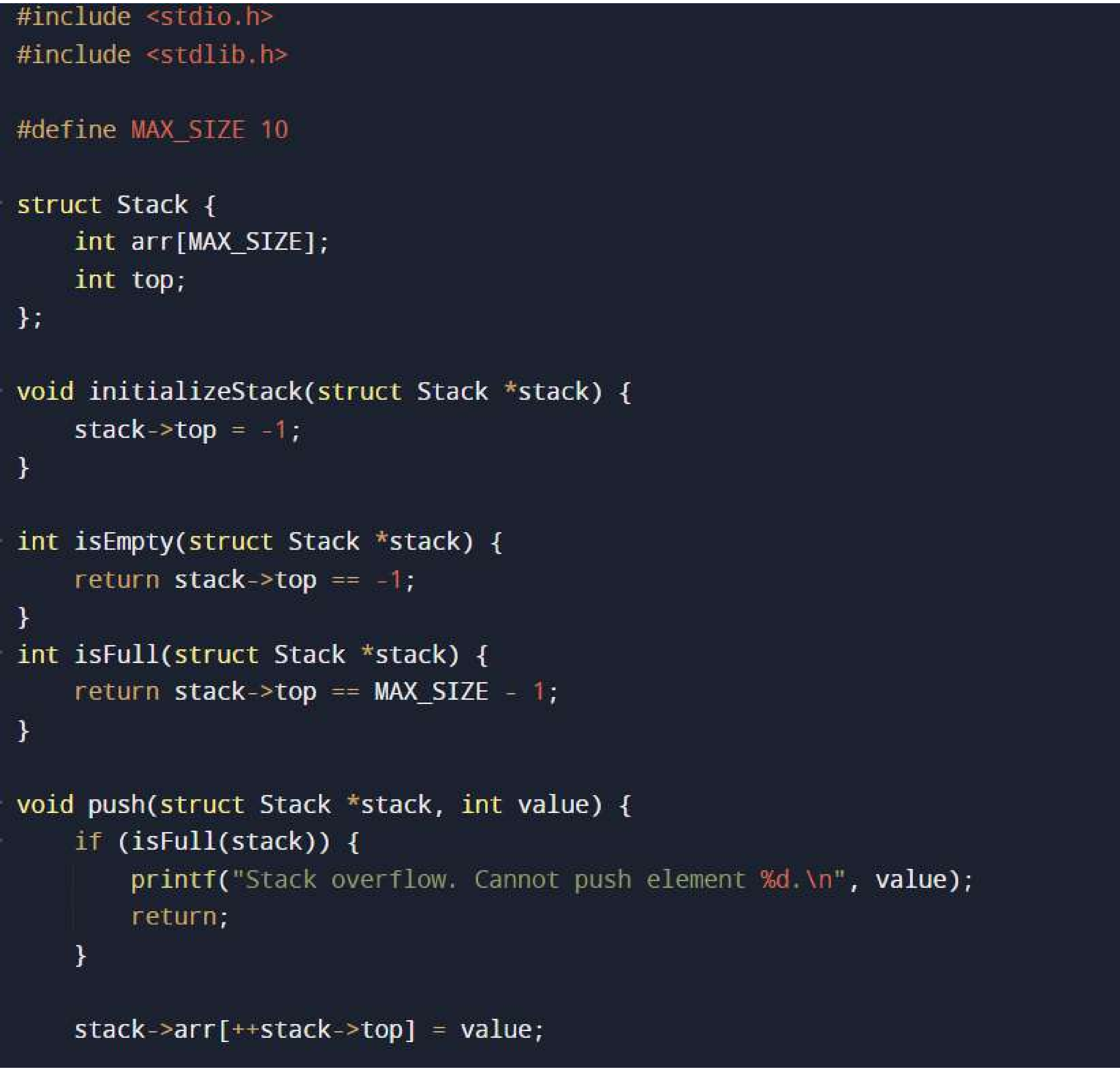
21/12



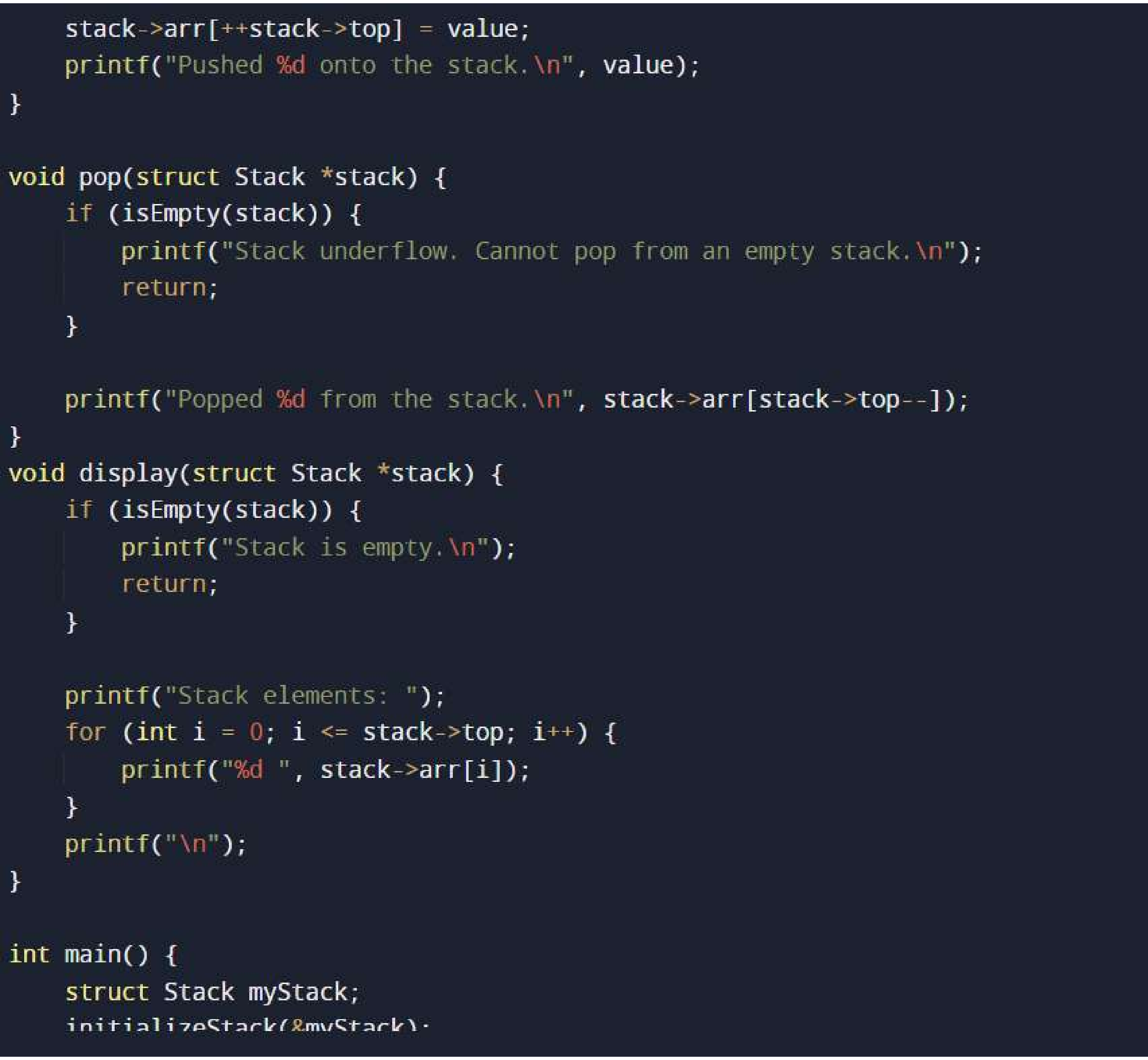
```c
#include <stdio.h>
void swap(int *a,int *b)
{

    int temp=*a;
    *a=*b;
    *b=temp;
}
int main()
{
    int num1,num2;
    printf("enter the first number\n");
    scanf("%d",&num1);
    printf("enter the second number\n");
    scanf("%d",&num2);
    printf("before swapping num1=%d,num2=%d\n",num1,num2);
    swap(&num1,&num2);
    printf("after swapping num1=%d,num2=%d\n",num1,num2);
    return(0);
}
```



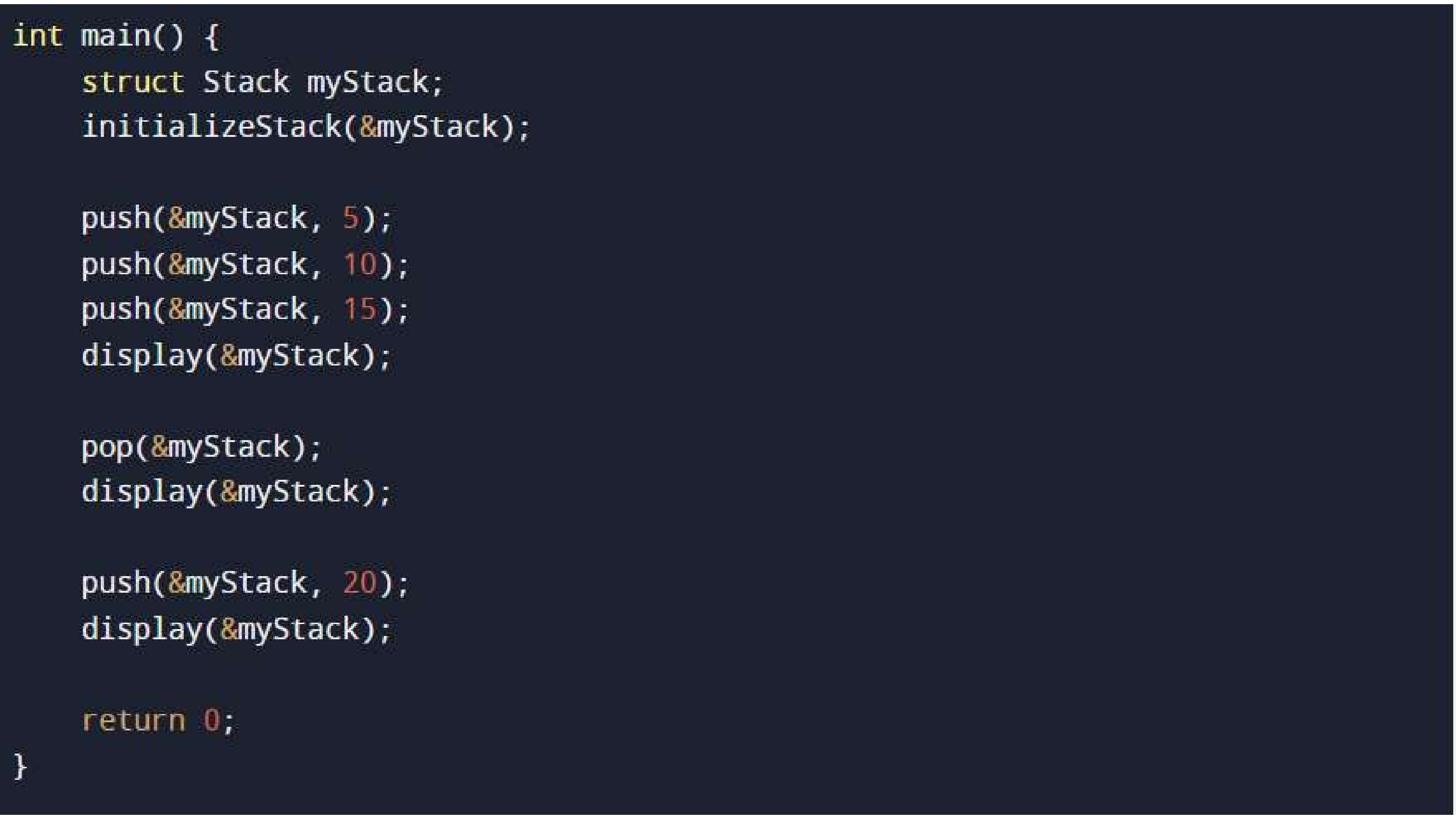
```
enter the first number
10
enter the second number
25
before swapping num1=10,num2=25
after swapping num1=25,num2=10
```



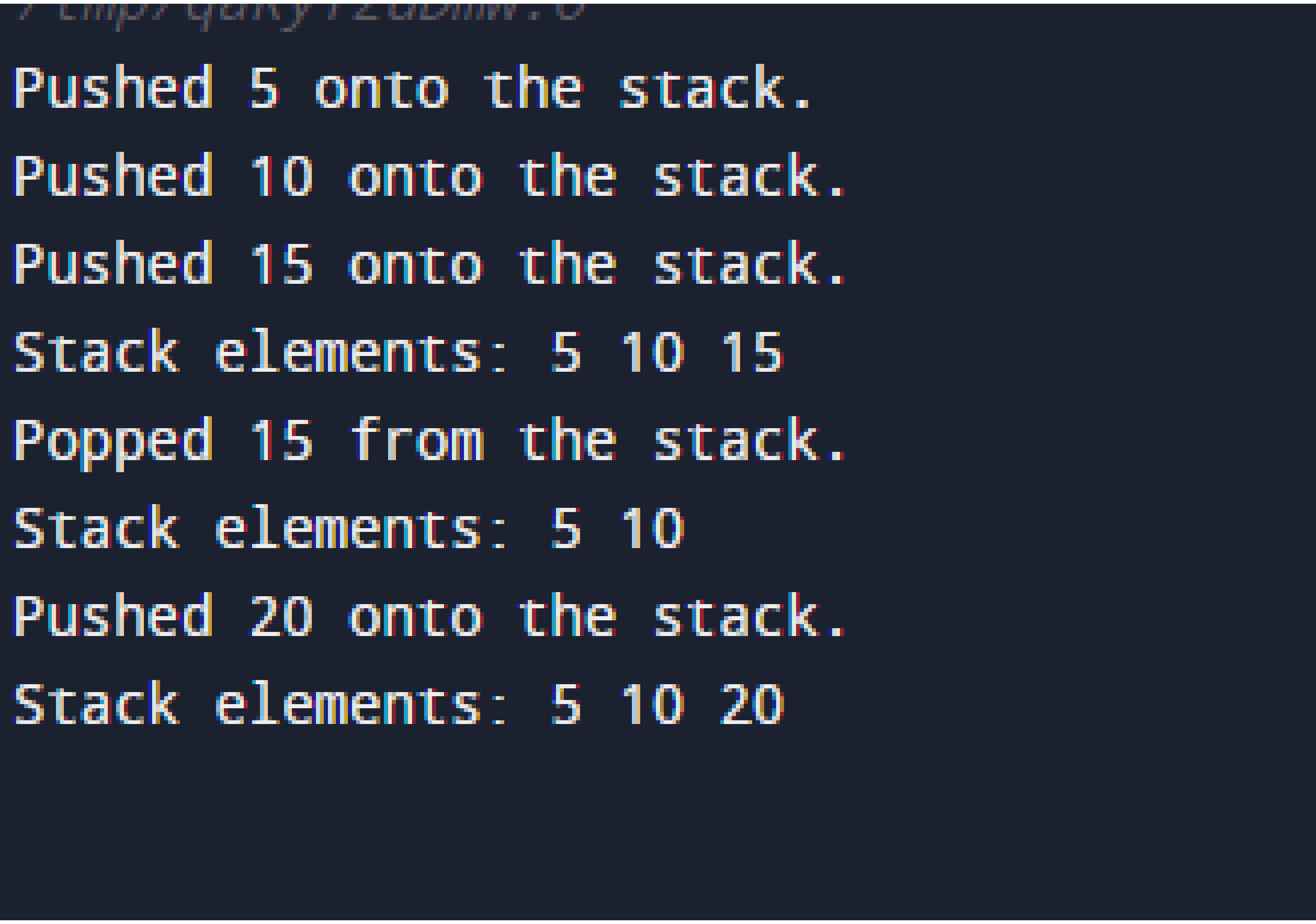
```
#include <stdio.h>
#include <stdlib.h>

#define MAX_SIZE 10

struct Stack {
    int arr[MAX_SIZE];
    int top;
};

void initializeStack(struct Stack *stack) {
    stack->top = -1;
}

int isEmpty(struct Stack *stack) {
    return stack->top == -1;
}
int isFull(struct Stack *stack) {
    return stack->top == MAX_SIZE - 1;
}

void push(struct Stack *stack, int value) {
    if (isFull(stack)) {
        printf("Stack overflow. Cannot push element %d.\n", value);
        return;
    }

    stack->arr[++stack->top] = value;
```



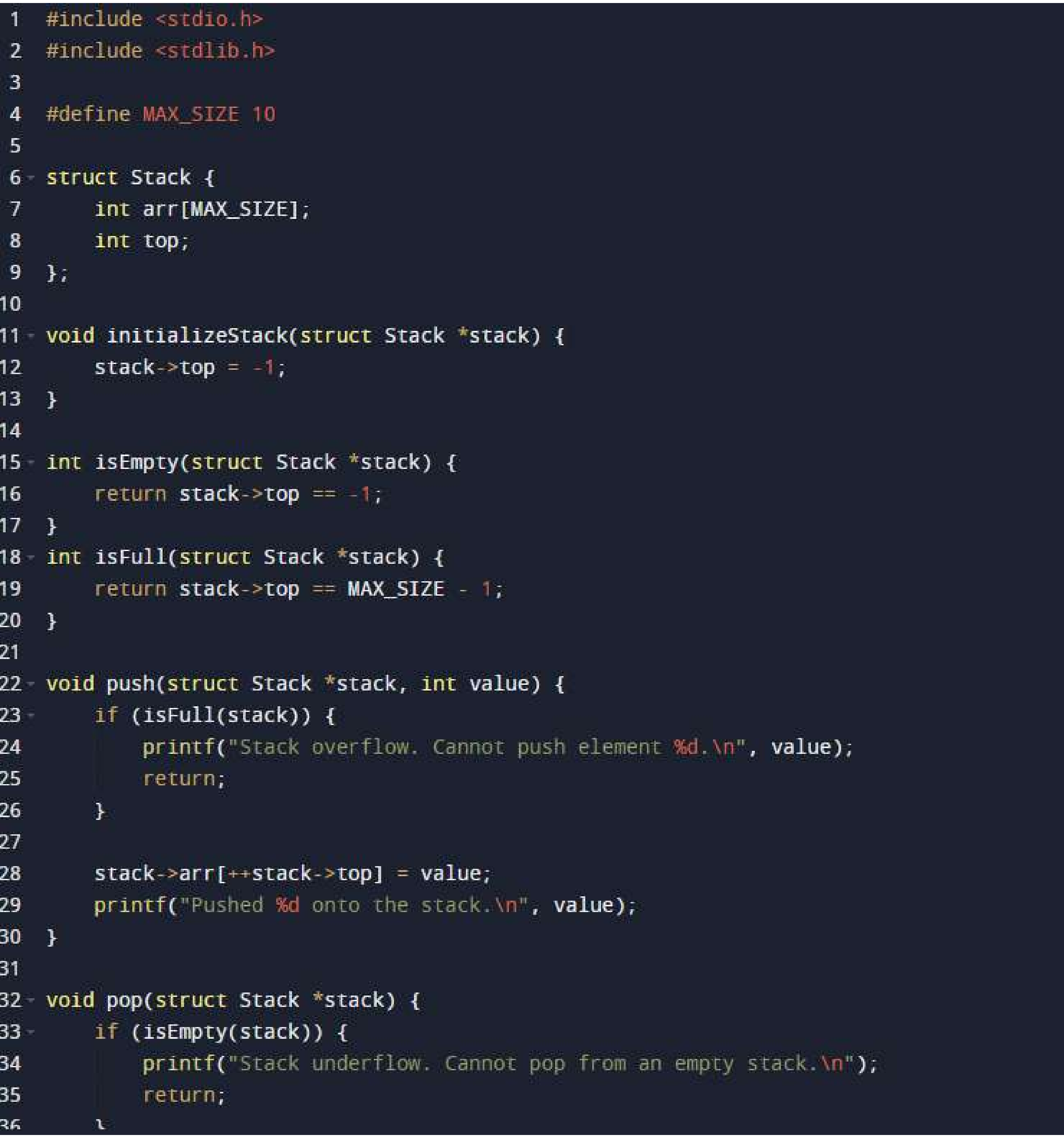
```
    stack->arr[++stack->top] = value;
    printf("Pushed %d onto the stack.\n", value);
}

void pop(struct Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack underflow. Cannot pop from an empty stack.\n");
        return;
    }

    printf("Popped %d from the stack.\n", stack->arr[stack->top--]);
}
void display(struct Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty.\n");
        return;
    }

    printf("Stack elements: ");
    for (int i = 0; i <= stack->top; i++) {
        printf("%d ", stack->arr[i]);
    }
    printf("\n");
}

int main() {
    struct Stack myStack;
    initializeStack(&myStack);
```



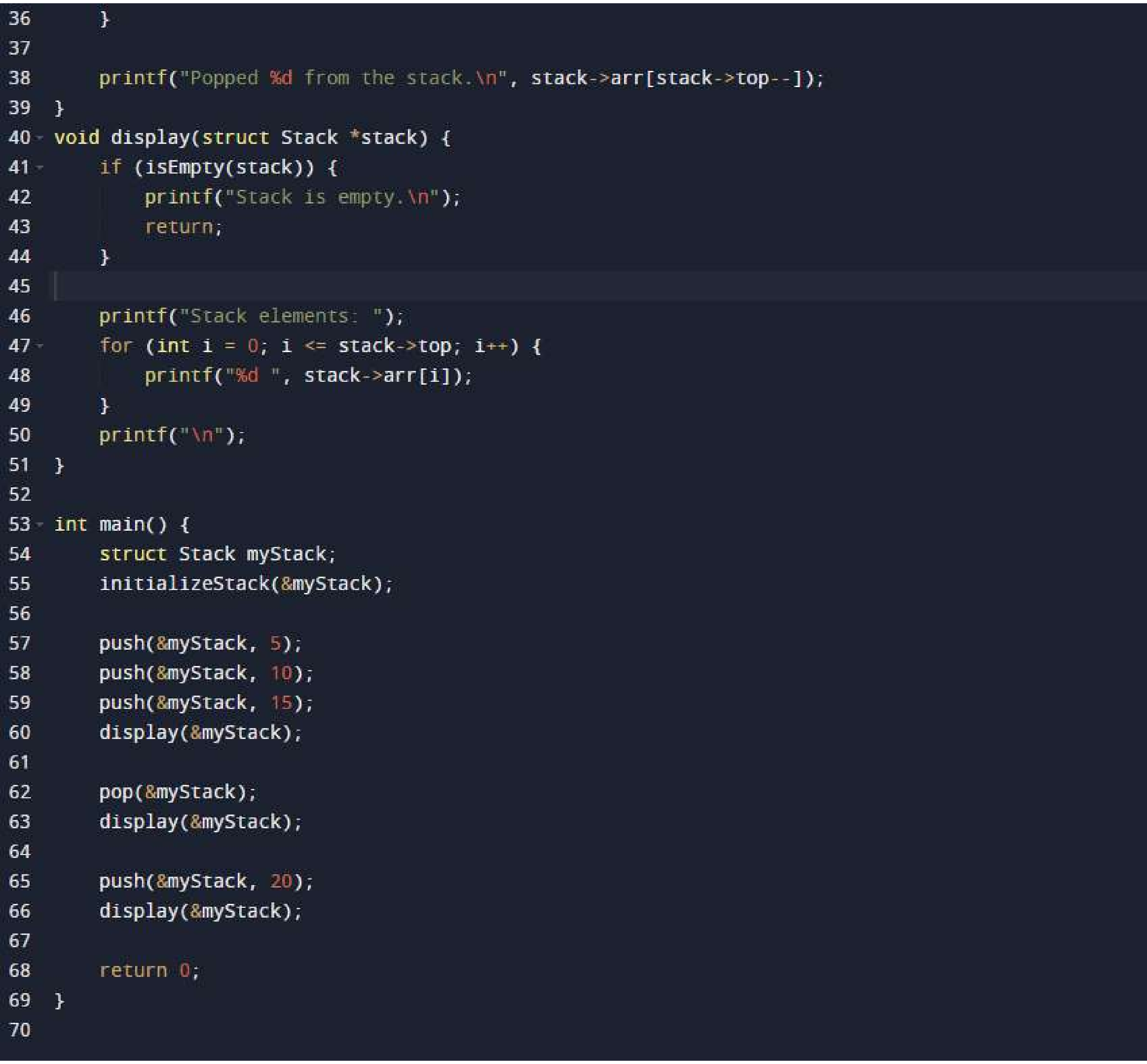
```c
int main() {
    struct Stack myStack;
    initializeStack(&myStack);

    push(&myStack, 5);
    push(&myStack, 10);
    push(&myStack, 15);
    display(&myStack);

    pop(&myStack);
    display(&myStack);

    push(&myStack, 20);
    display(&myStack);

    return 0;
}
```



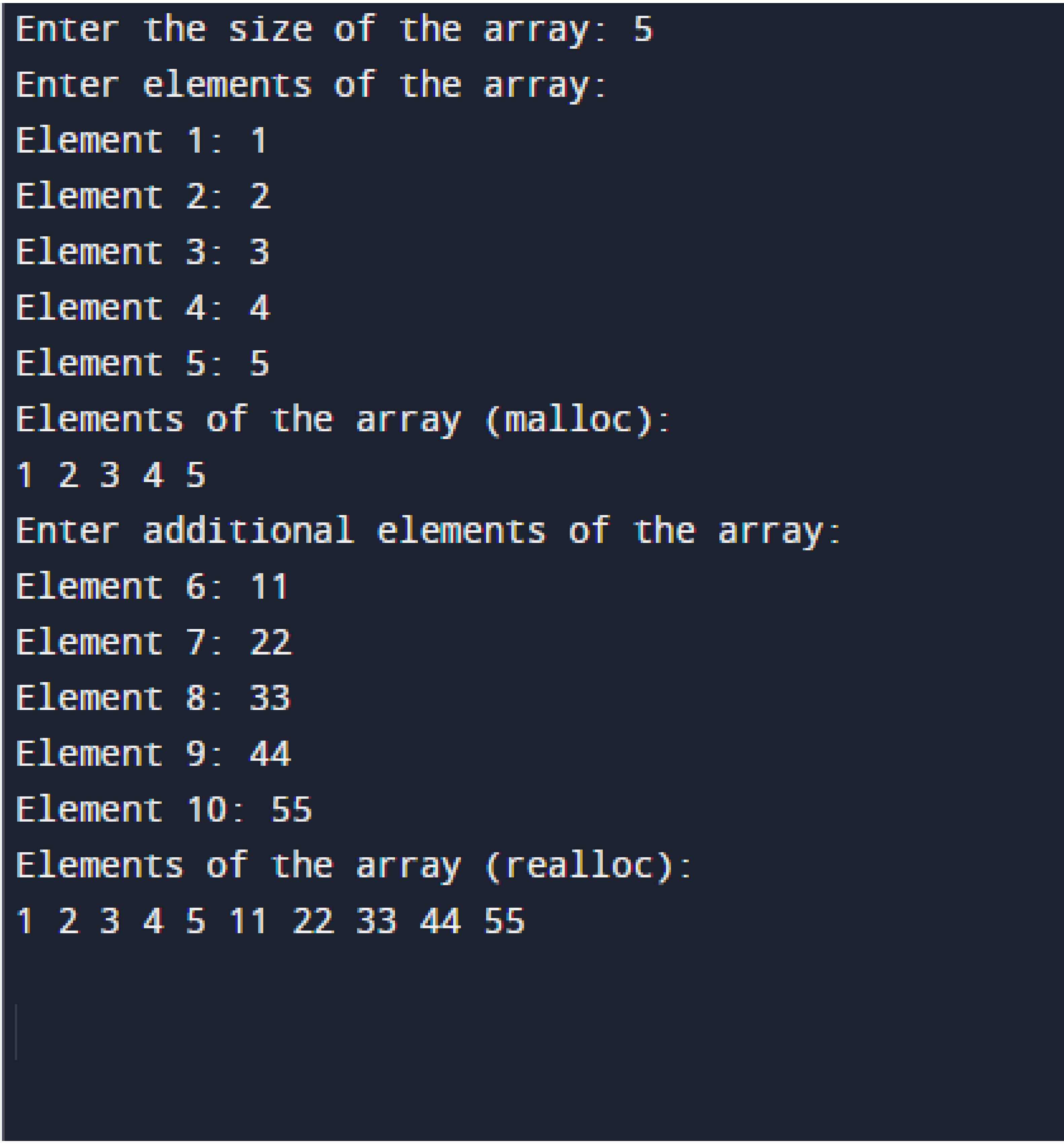
```
/tmp/qaKy1Zubmm.o
Pushed 5 onto the stack.
Pushed 10 onto the stack.
Pushed 15 onto the stack.
Stack elements: 5 10 15
Popped 15 from the stack.
Stack elements: 5 10
Pushed 20 onto the stack.
Stack elements: 5 10 20
```



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

#define MAX_SIZE 10

struct Stack {
    int arr[MAX_SIZE];
    int top;
};

void initializeStack(struct Stack *stack) {
    stack->top = -1;
}

int isEmpty(struct Stack *stack) {
    return stack->top == -1;
}
int isFull(struct Stack *stack) {
    return stack->top == MAX_SIZE - 1;
}

void push(struct Stack *stack, int value) {
    if (isFull(stack)) {
        printf("Stack overflow. Cannot push element %d.\n", value);
        return;
    }

    stack->arr[++stack->top] = value;
    printf("Pushed %d onto the stack.\n", value);
}

void pop(struct Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack underflow. Cannot pop from an empty stack.\n");
        return;
    }
```



```c
    }

    printf("Popped %d from the stack.\n", stack->arr[stack->top--]);
}
void display(struct Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty.\n");
        return;
    }

    printf("Stack elements: ");
    for (int i = 0; i <= stack->top; i++) {
        printf("%d ", stack->arr[i]);
    }
    printf("\n");
}

int main() {
    struct Stack myStack;
    initializeStack(&myStack);

    push(&myStack, 5);
    push(&myStack, 10);
    push(&myStack, 15);
    display(&myStack);

    pop(&myStack);
    display(&myStack);

    push(&myStack, 20);
    display(&myStack);

    return 0;
}

```



```
Enter the size of the array: 5
Enter elements of the array:
Element 1: 1
Element 2: 2
Element 3: 3
Element 4: 4
Element 5: 5
Elements of the array (malloc):
1 2 3 4 5
Enter additional elements of the array:
Element 6: 11
Element 7: 22
Element 8: 33
Element 9: 44
Element 10: 55
Elements of the array (realloc):
1 2 3 4 5 11 22 33 44 55
```