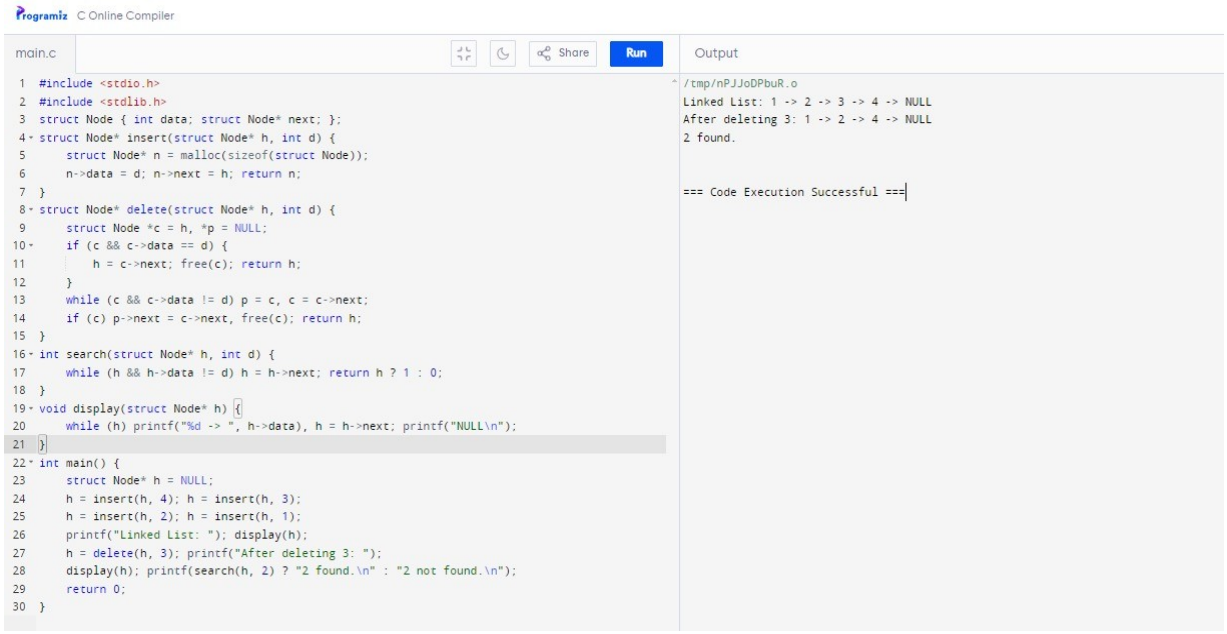
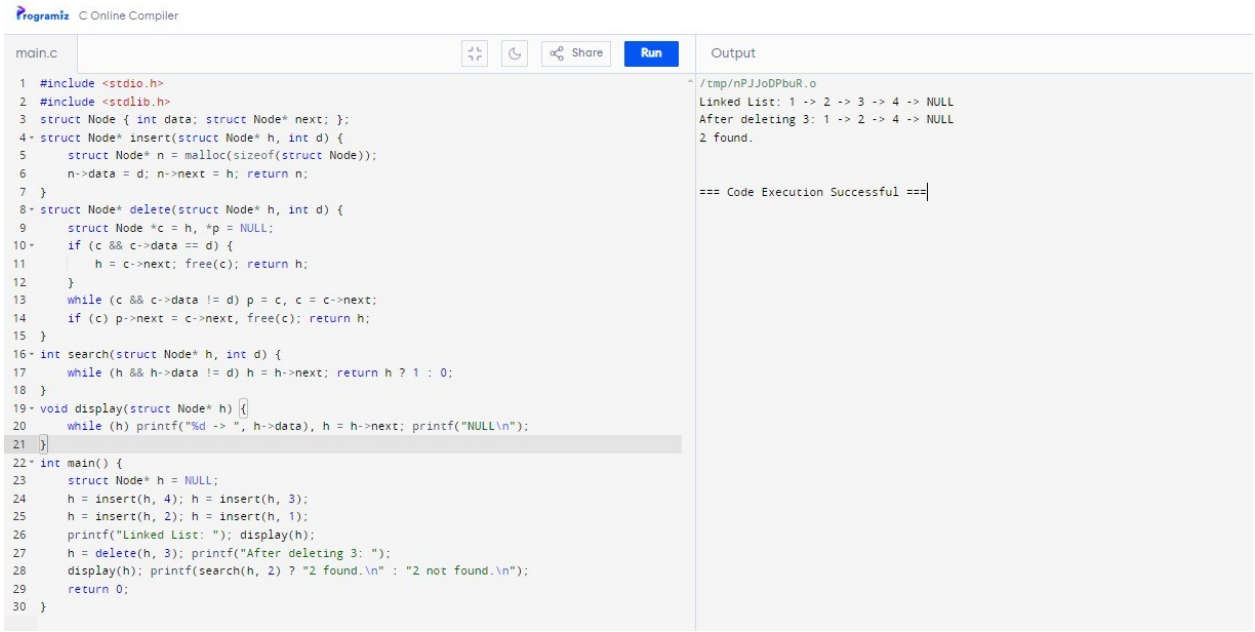
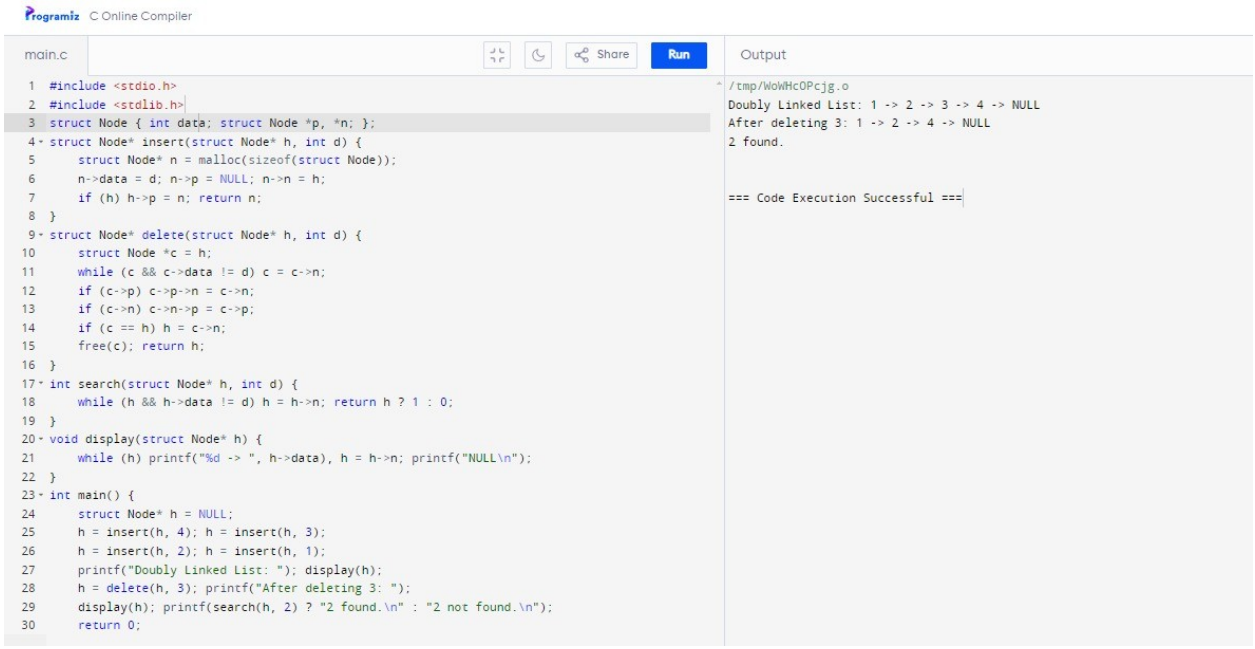
1) Write a c program for linked list.



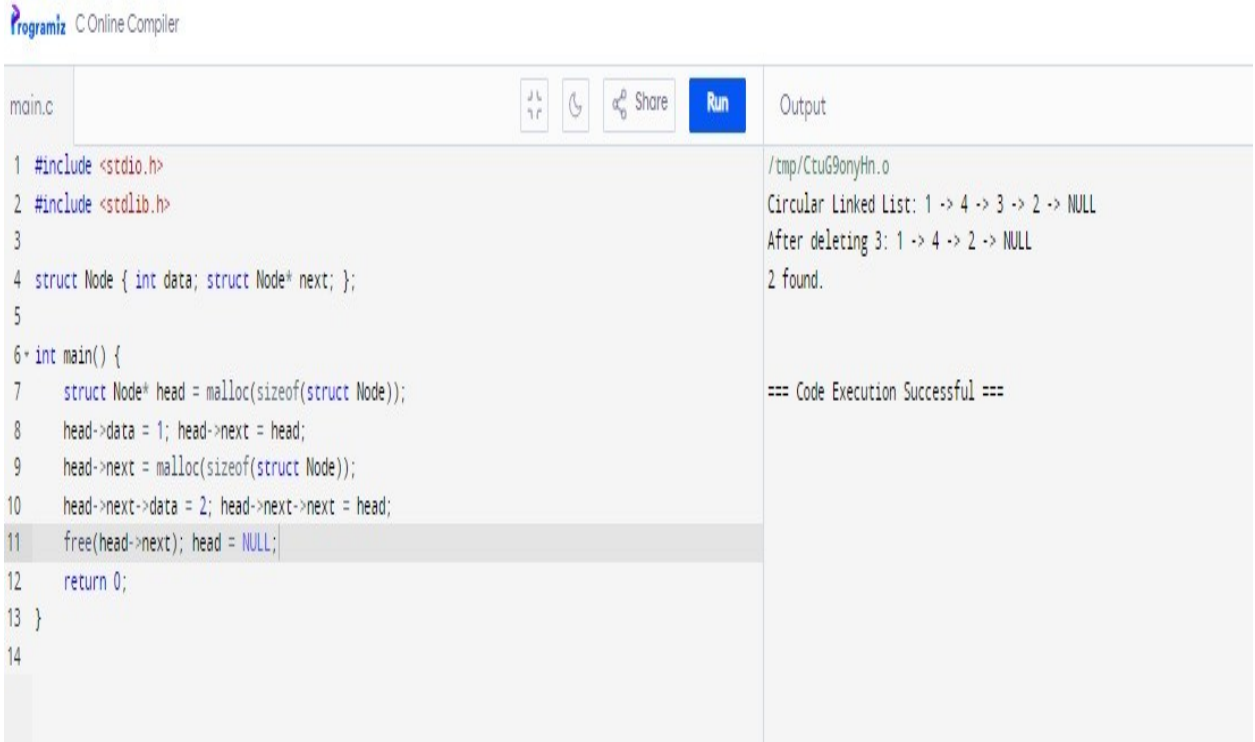
2) Write a c program for single linked list.



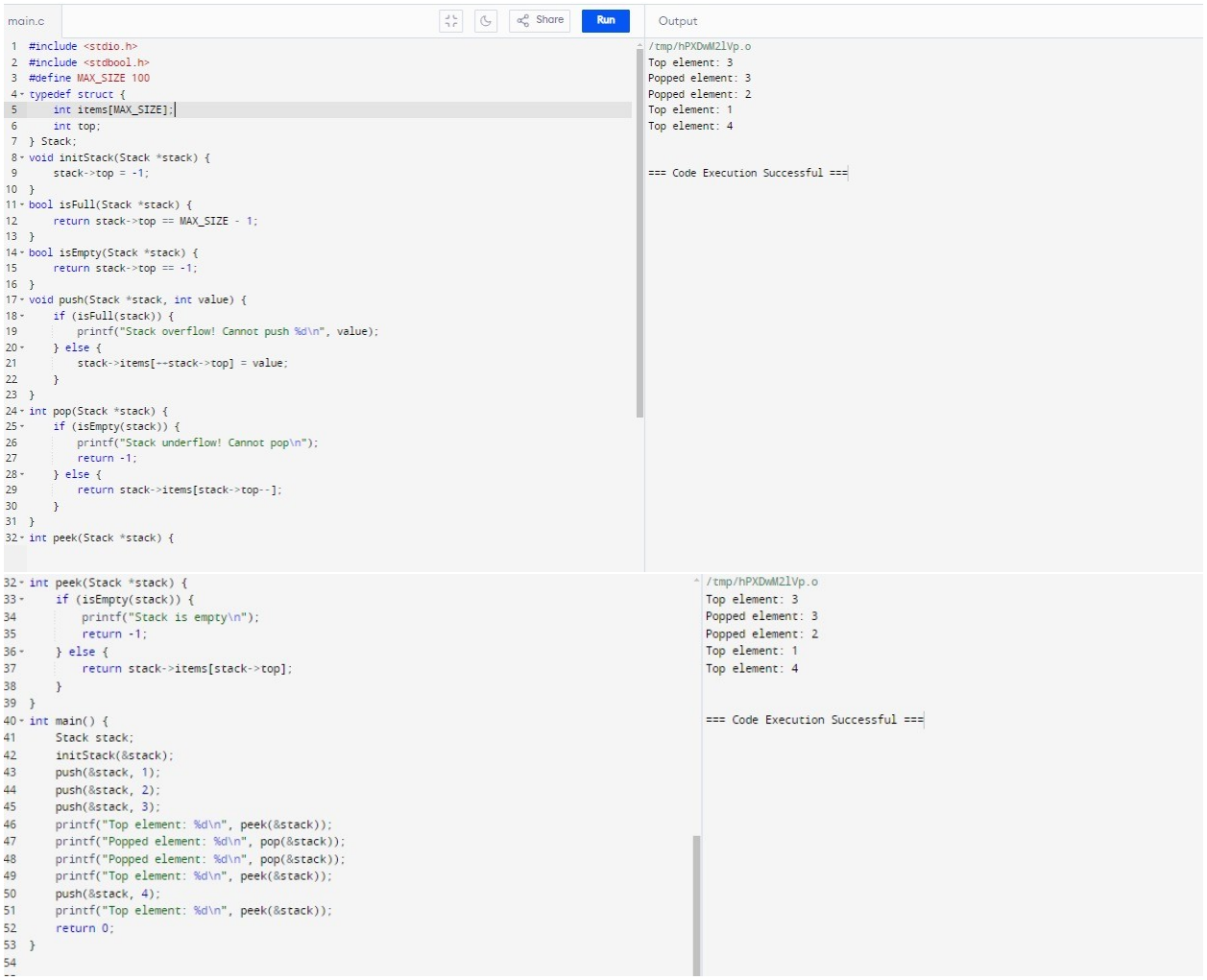
3) Write a c program for double linked list.



4) Write a c program circular linked list.



5) Write a c program stack using all operations push, pop, peek, full, empty.

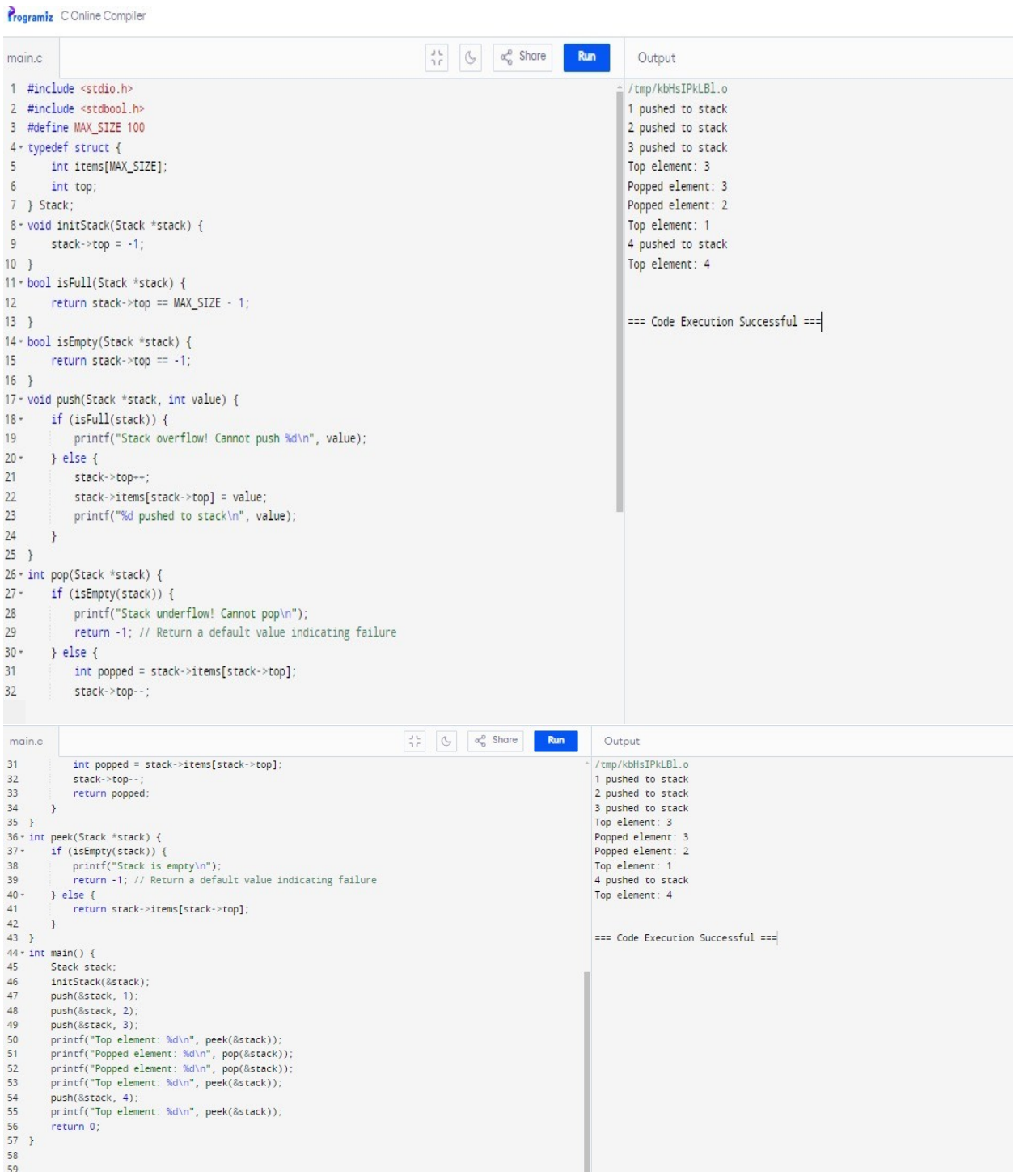


```c
#include <stdio.h>
#include <stdbool.h>
#define MAX_SIZE 100
typedef struct {
    int items[MAX_SIZE];
    int top;
} Stack;
void initStack(Stack *stack) {
    stack->top = -1;
}
bool isFull(Stack *stack) {
    return stack->top == MAX_SIZE - 1;
}
bool isEmpty(Stack *stack) {
    return stack->top == -1;
}
void push(Stack *stack, int value) {
    if (isFull(stack)) {
        printf("Stack overflow! Cannot push %d\n", value);
    } else {
        stack->items[++stack->top] = value;
    }
}
int pop(Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack underflow! Cannot pop\n");
        return -1;
    } else {
        return stack->items[stack->top--];
    }
}
int peek(Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty\n");
        return -1;
    } else {
        return stack->items[stack->top];
    }
}
int main() {
    Stack stack;
    initStack(&stack);
    push(&stack, 1);
    push(&stack, 2);
    push(&stack, 3);
    printf("Top element: %d\n", peek(&stack));
    printf("Popped element: %d\n", pop(&stack));
    printf("Popped element: %d\n", pop(&stack));
    printf("Top element: %d\n", peek(&stack));
    push(&stack, 4);
    printf("Top element: %d\n", peek(&stack));
    return 0;
}
```

```
/tmp/hPXDwM2lVp.o
Top element: 3
Popped element: 3
Popped element: 2
Top element: 1
Top element: 4


=== Code Execution Successful ===
```

6) Write a c program for array implementation of stack.



7) Write a c program that implements a stack using linked list.

Programiz C Online Compiler

main.c

Share Run

```c
#include <stdio.h>
#include <stdbool.h>
#define MAX_SIZE 100
typedef struct {
    int items[MAX_SIZE];
    int top;
} Stack;
void initStack(Stack *stack) {
    stack->top = -1;
}
bool isFull(Stack *stack) {
    return stack->top == MAX_SIZE - 1;
}
bool isEmpty(Stack *stack) {
    return stack->top == -1;
}
void push(Stack *stack, int value) {
    if (isFull(stack)) {
        printf("Stack overflow! Cannot push %d\n", value);
    } else {
        stack->top++;
        stack->items[stack->top] = value;
        printf("%d pushed to stack\n", value);
    }
}
int pop(Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack underflow! Cannot pop\n");
        return -1; // Return a default value indicating failure
    } else {
        int popped = stack->items[stack->top];
        stack->top--;
        int popped = stack->items[stack->top];
        stack->top--;
        return popped;
    }
}
int peek(Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty\n");
        return -1; // Return a default value indicating failure
    } else {
        return stack->items[stack->top];
    }
}
int main() {
    Stack stack;
    initStack(&stack);
    push(&stack, 1);
    push(&stack, 2);
    push(&stack, 3);
    printf("Top element: %d\n", peek(&stack));
    printf("Popped element: %d\n", pop(&stack));
    printf("Popped element: %d\n", pop(&stack));
    printf("Top element: %d\n", peek(&stack));
    push(&stack, 4);
    printf("Top element: %d\n", peek(&stack));
    return 0;
}
```

Output

```
/tmp/kbHsIPkLBl.o
1 pushed to stack
2 pushed to stack
3 pushed to stack
Top element: 3
Popped element: 3
Popped element: 2
Top element: 1
4 pushed to stack
Top element: 4


=== Code Execution Successful ===
```