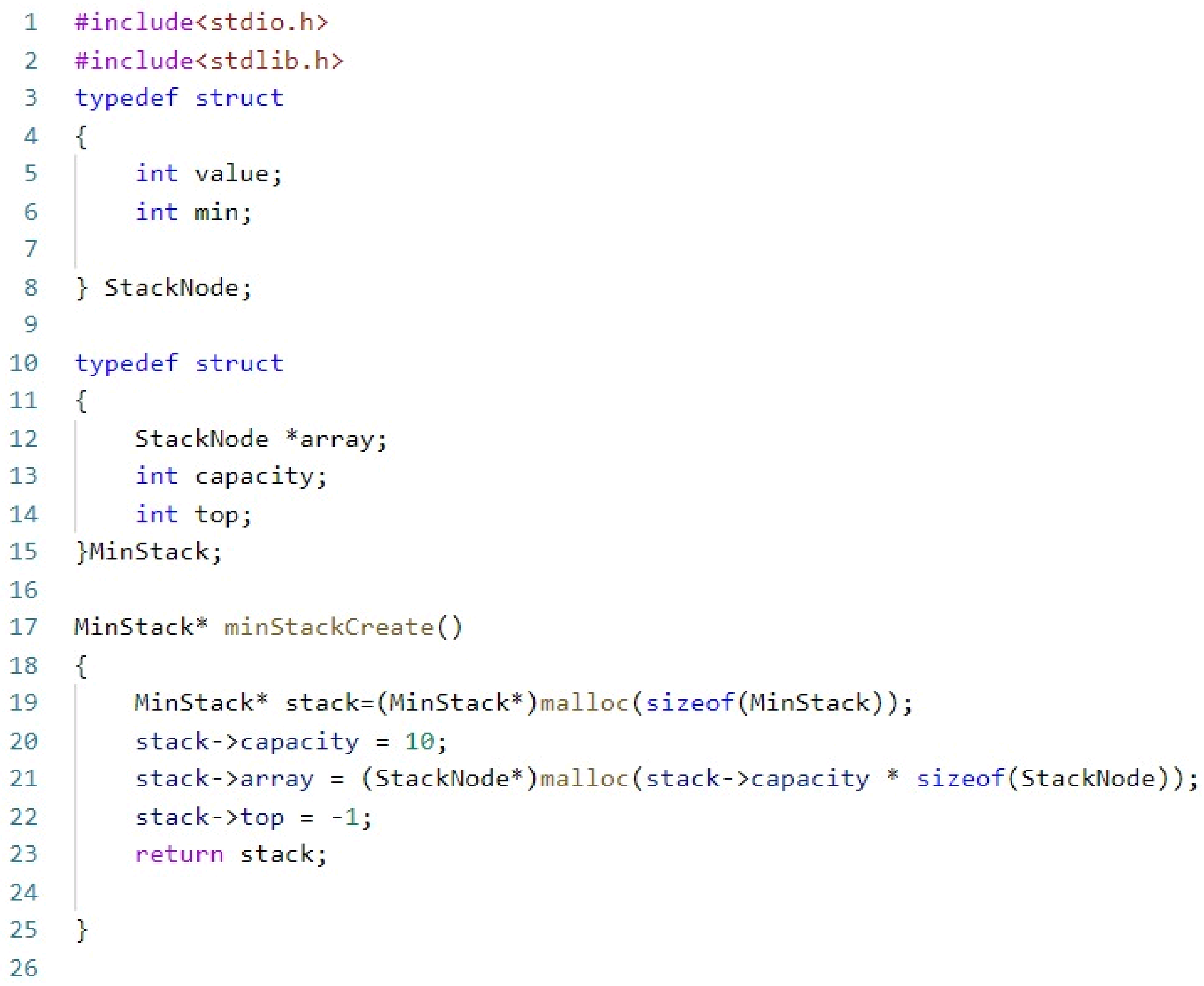
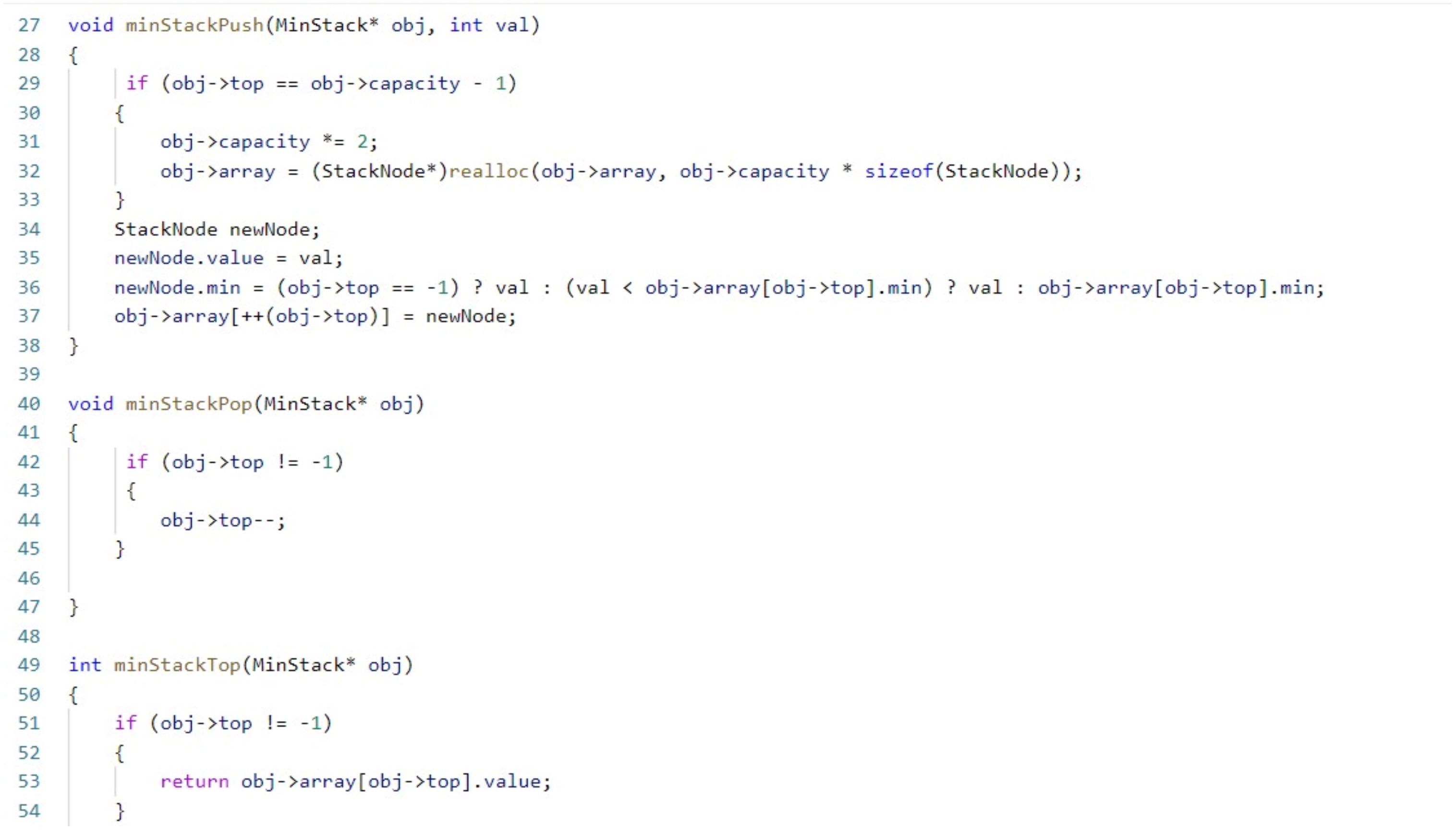
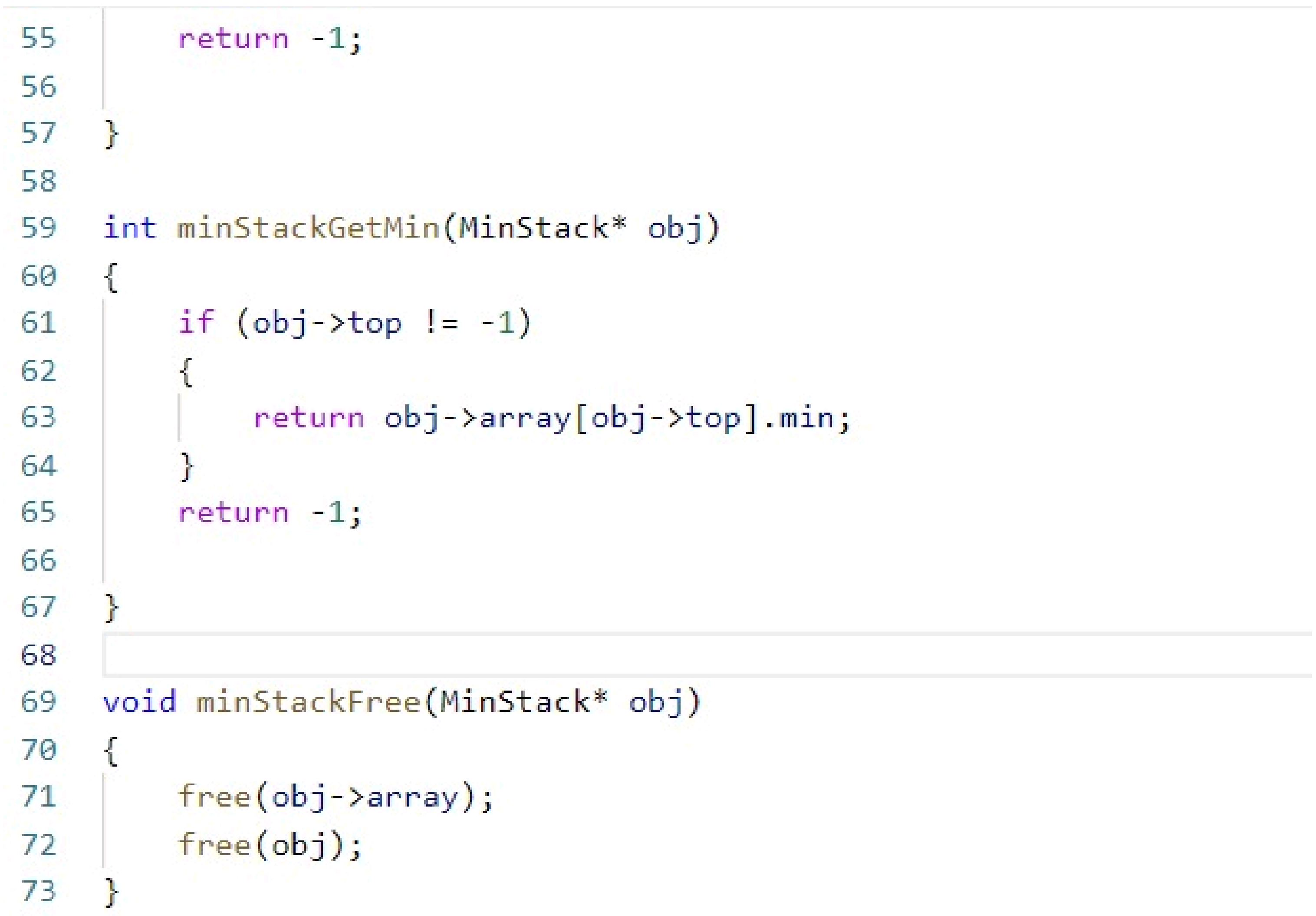


```c
#include<stdio.h>
#include<stdlib.h>
typedef struct
{
    int value;
    int min;

} StackNode;

typedef struct
{
    StackNode *array;
    int capacity;
    int top;
}MinStack;

MinStack* minStackCreate()
{
    MinStack* stack=(MinStack*)malloc(sizeof(MinStack));
    stack->capacity = 10;
    stack->array = (StackNode*)malloc(stack->capacity * sizeof(StackNode));
    stack->top = -1;
    return stack;

}

```



```c
void minStackPush(MinStack* obj, int val)
{
    if (obj->top == obj->capacity - 1)
    {
        obj->capacity *= 2;
        obj->array = (StackNode*)realloc(obj->array, obj->capacity * sizeof(StackNode));
    }
    StackNode newNode;
    newNode.value = val;
    newNode.min = (obj->top == -1) ? val : (val < obj->array[obj->top].min) ? val : obj->array[obj->top].min;
    obj->array[++(obj->top)] = newNode;
}

void minStackPop(MinStack* obj)
{
    if (obj->top != -1)
    {
        obj->top--;
    }

}

int minStackTop(MinStack* obj)
{
    if (obj->top != -1)
    {
        return obj->array[obj->top].value;
    }
```



```c
    return -1;

}

int minStackGetMin(MinStack* obj)
{
    if (obj->top != -1)
    {
        return obj->array[obj->top].min;
    }
    return -1;

}

void minStackFree(MinStack* obj)
{
    free(obj->array);
    free(obj);
}
```