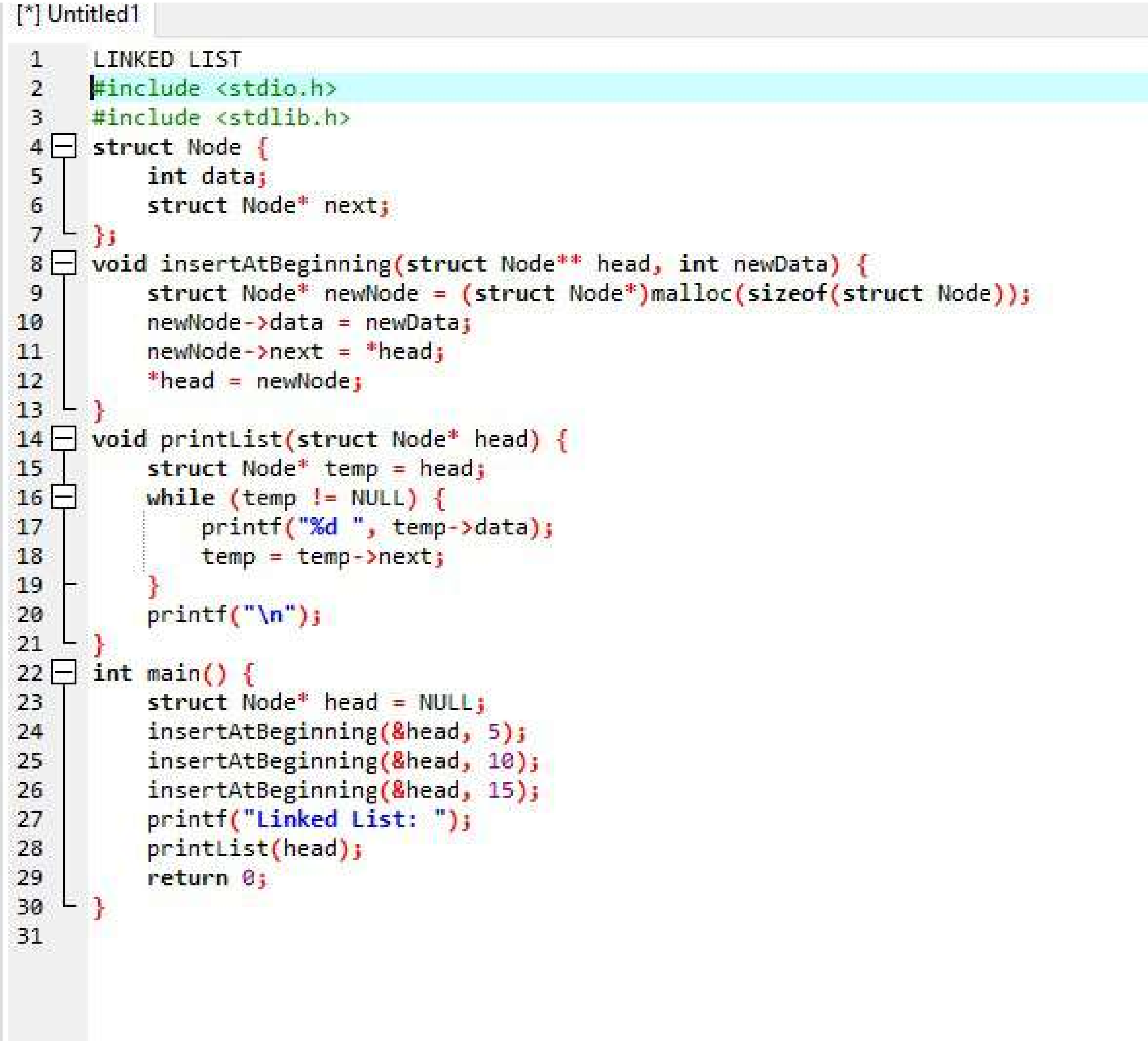
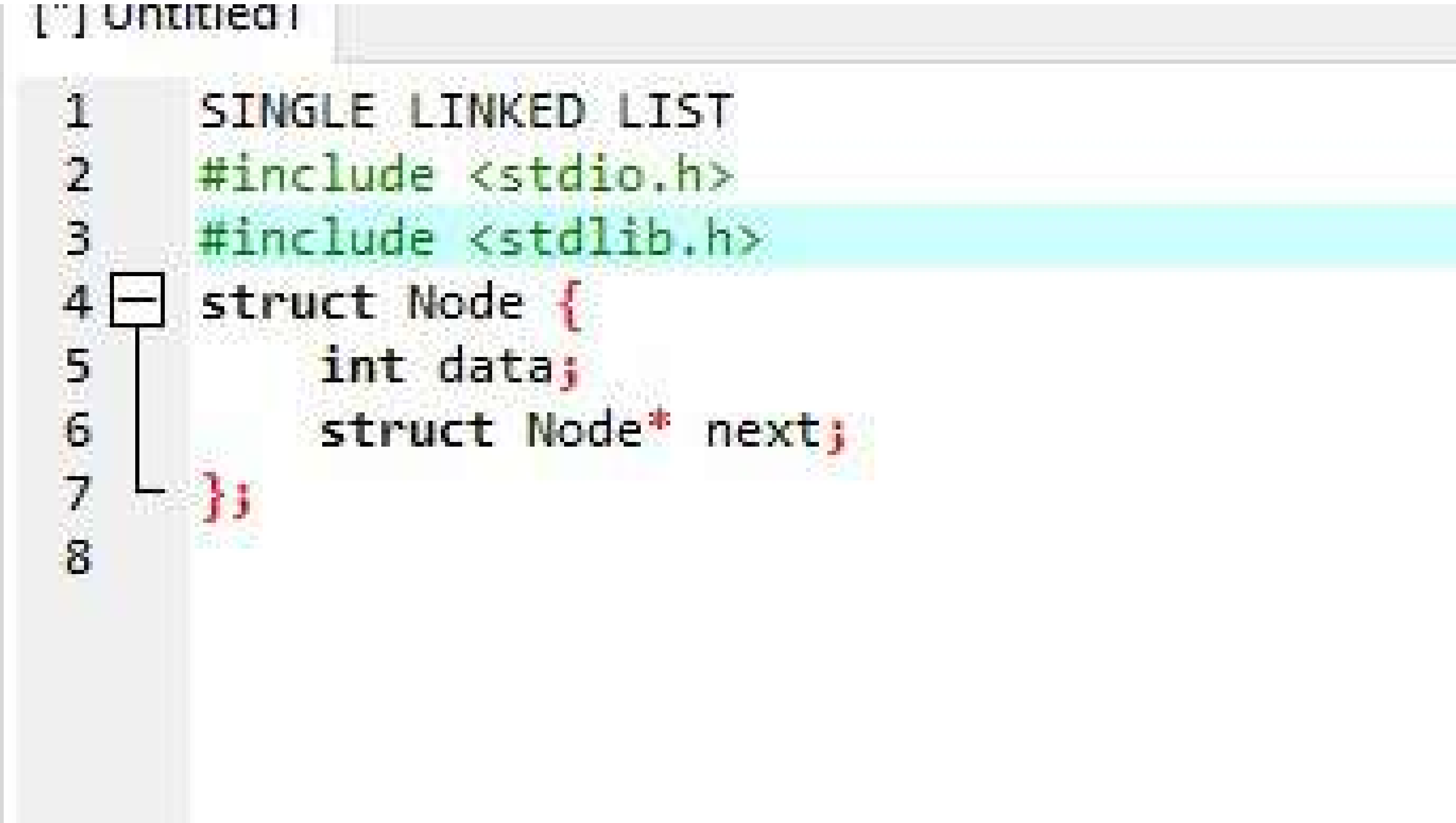
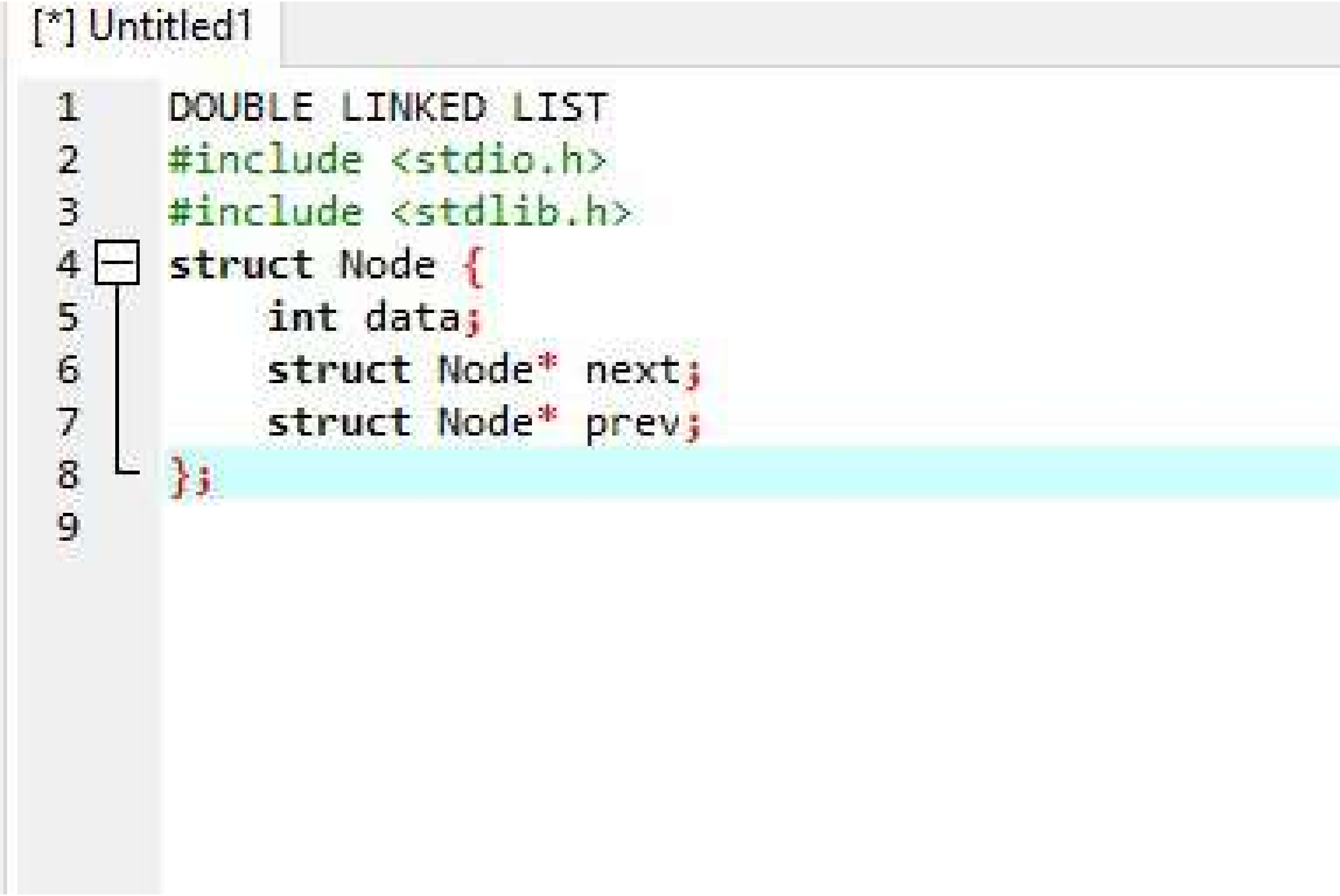


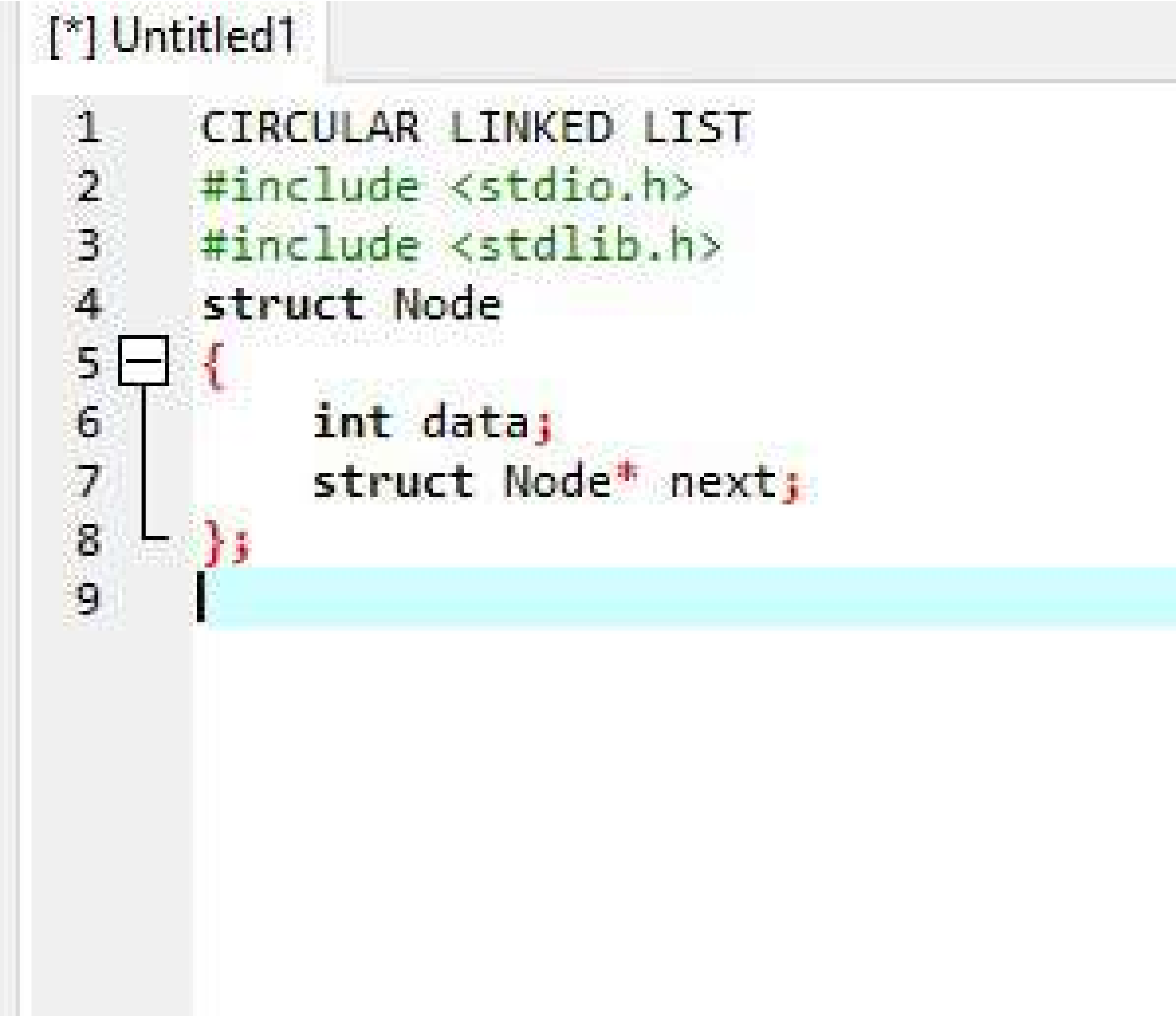
```
LINKED LIST
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
void insertAtBeginning(struct Node** head, int newData) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = newData;
    newNode->next = *head;
    *head = newNode;
}
void printList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
int main() {
    struct Node* head = NULL;
    insertAtBeginning(&head, 5);
    insertAtBeginning(&head, 10);
    insertAtBeginning(&head, 15);
    printf("Linked List: ");
    printList(head);
    return 0;
}
```



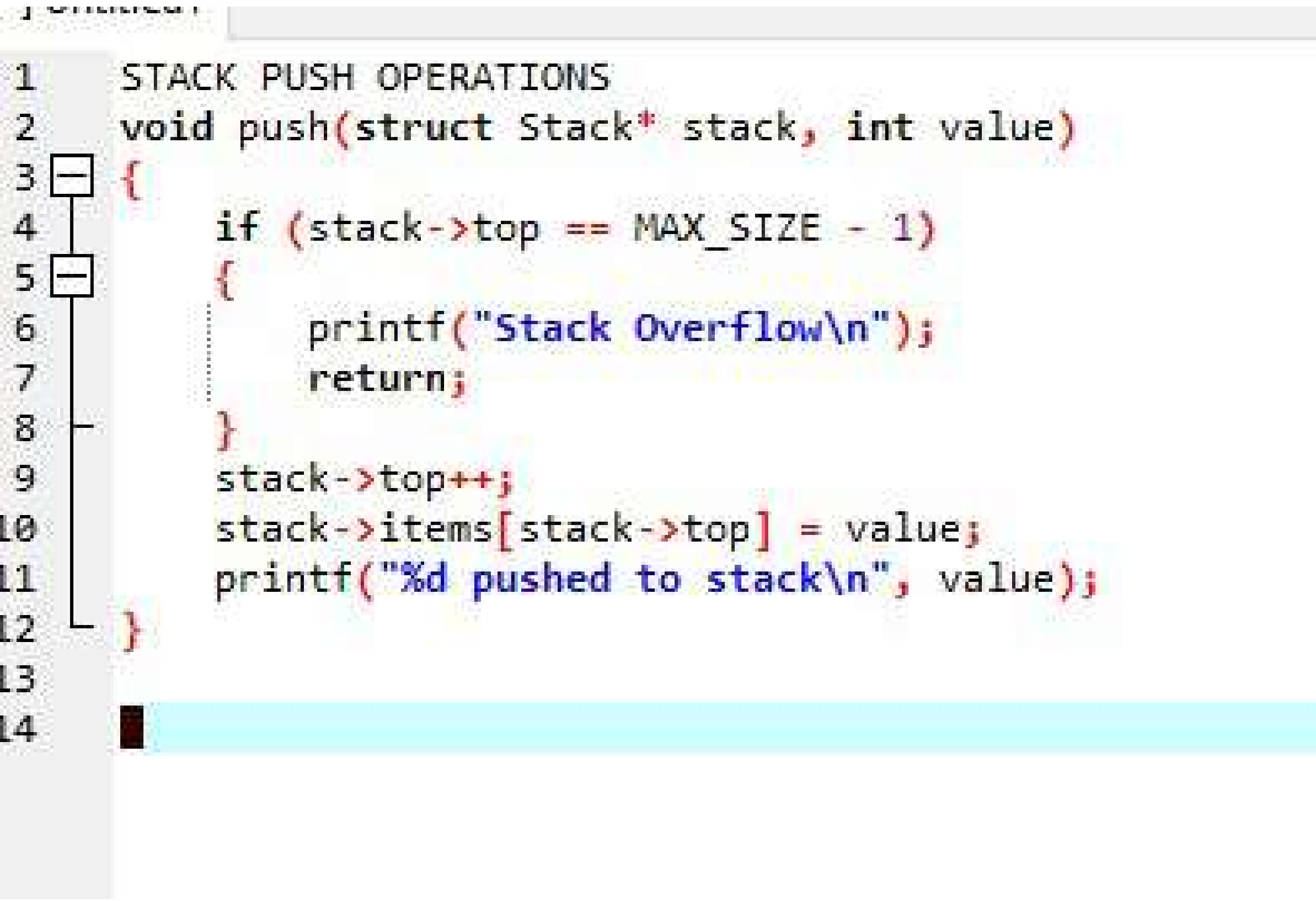
```
SINGLE LINKED LIST
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
```



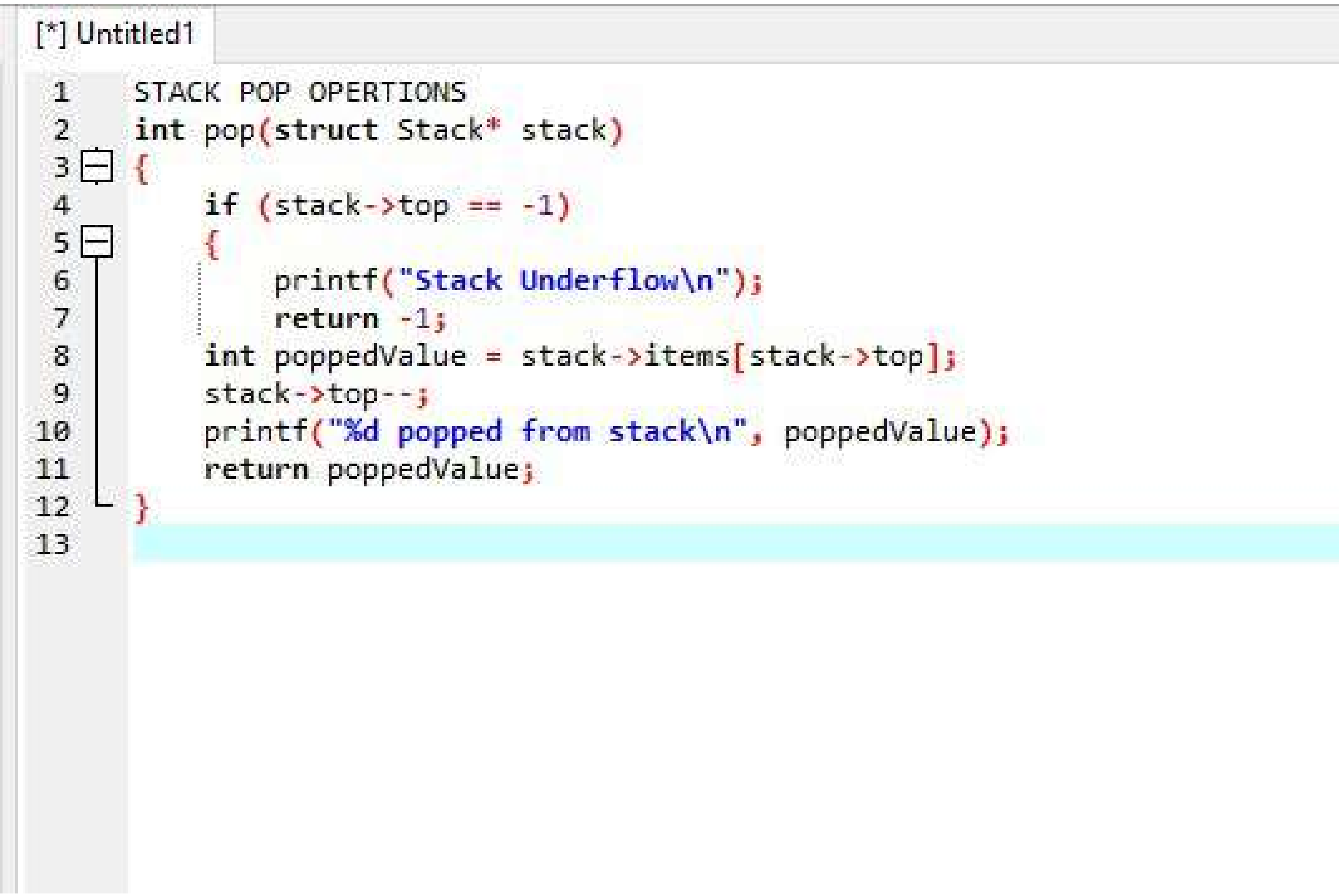
```
DOUBLE LINKED LIST
#include <stdio.h>
#include <stdlib.h>
struct Node {
     int data;
     struct Node* next;
     struct Node* prev;
};
```



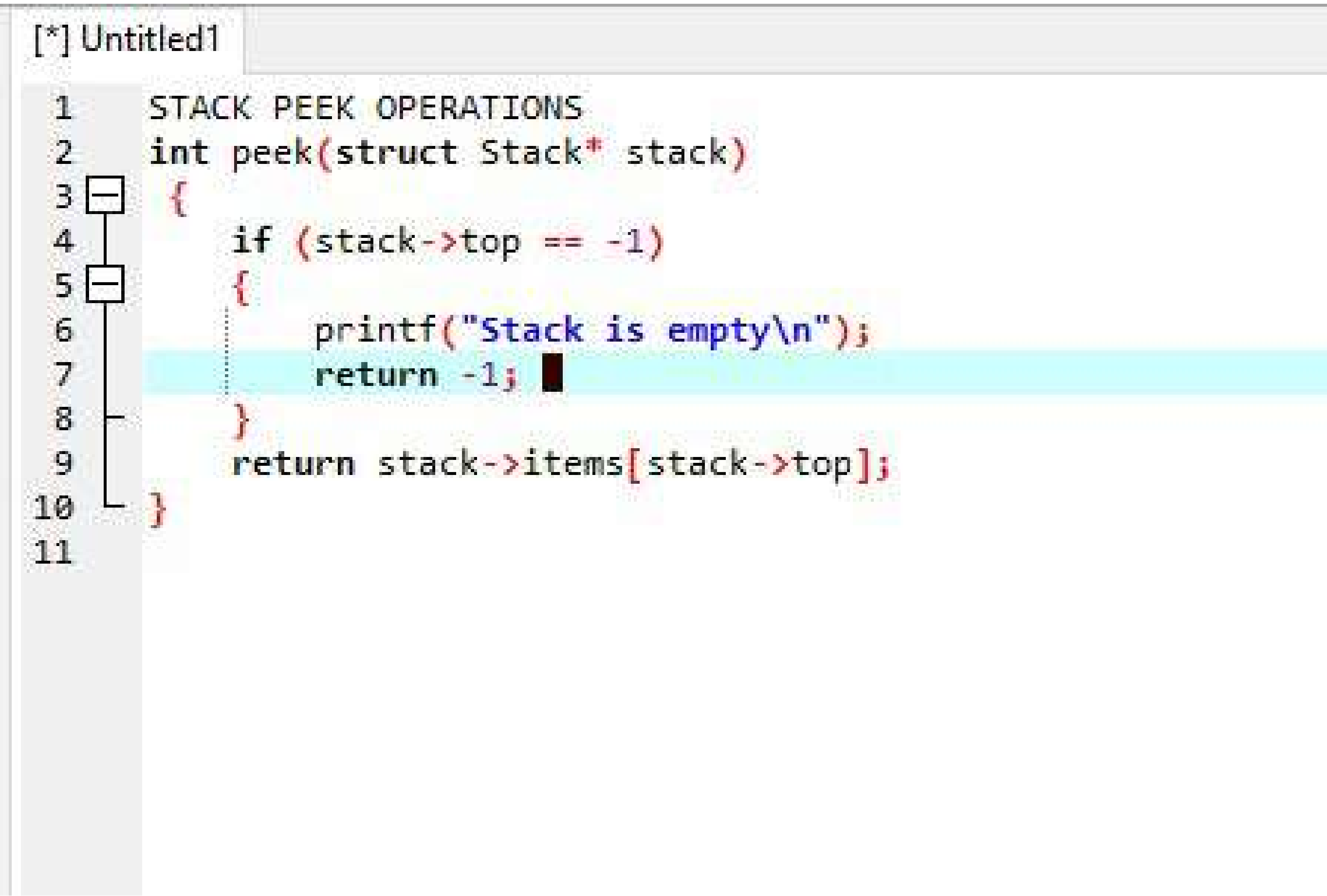
```
CIRCULAR LINKED LIST
#include <stdio.h>
#include <stdlib.h>
struct Node
{
	int data;
	struct Node* next;
};

```



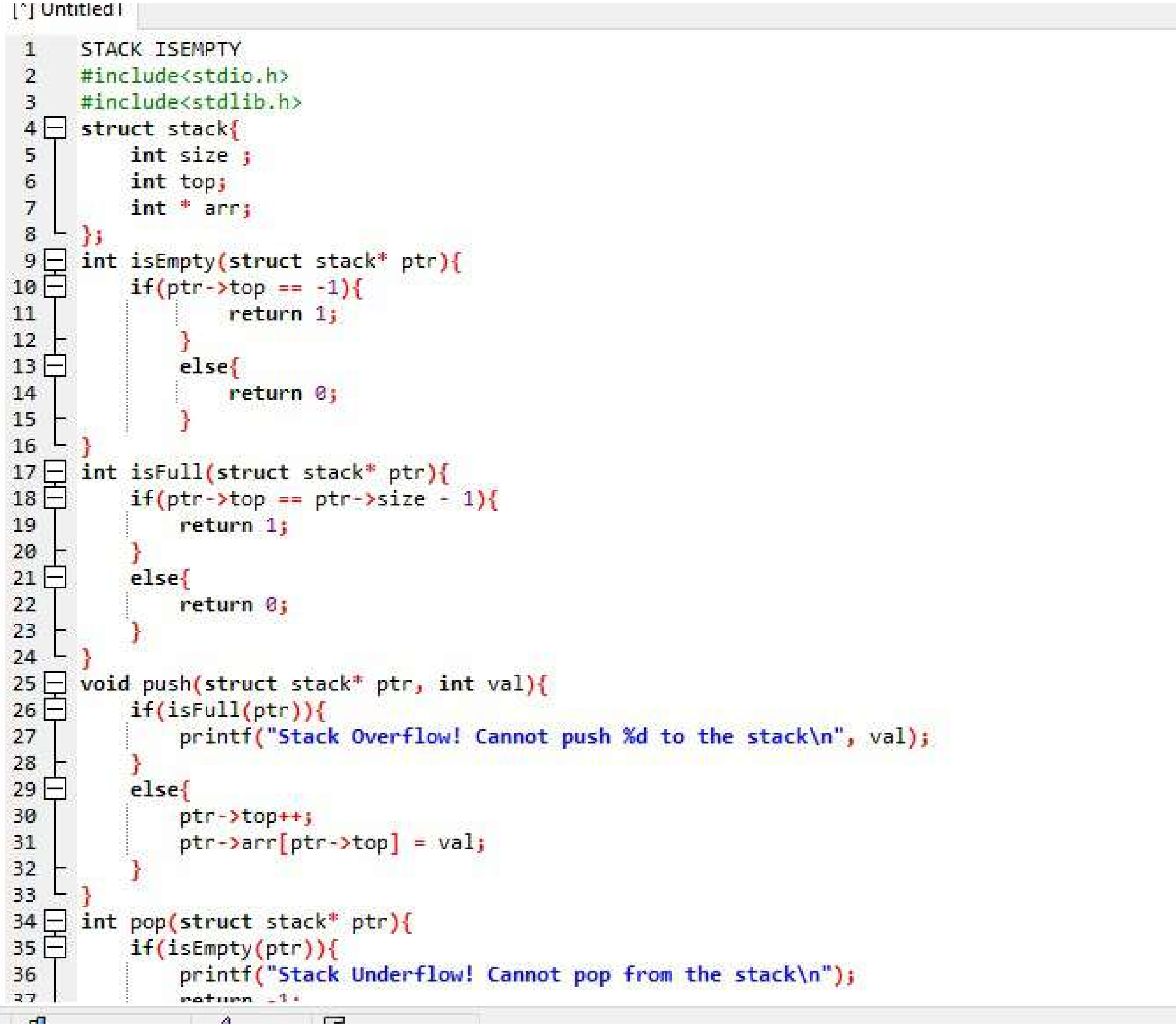
```
STACK PUSH OPERATIONS
void push(struct Stack* stack, int value)
{
    if (stack->top == MAX_SIZE - 1)
    {
        printf("Stack Overflow\n");
        return;
    }
    stack->top++;
    stack->items[stack->top] = value;
    printf("%d pushed to stack\n", value);
}
```



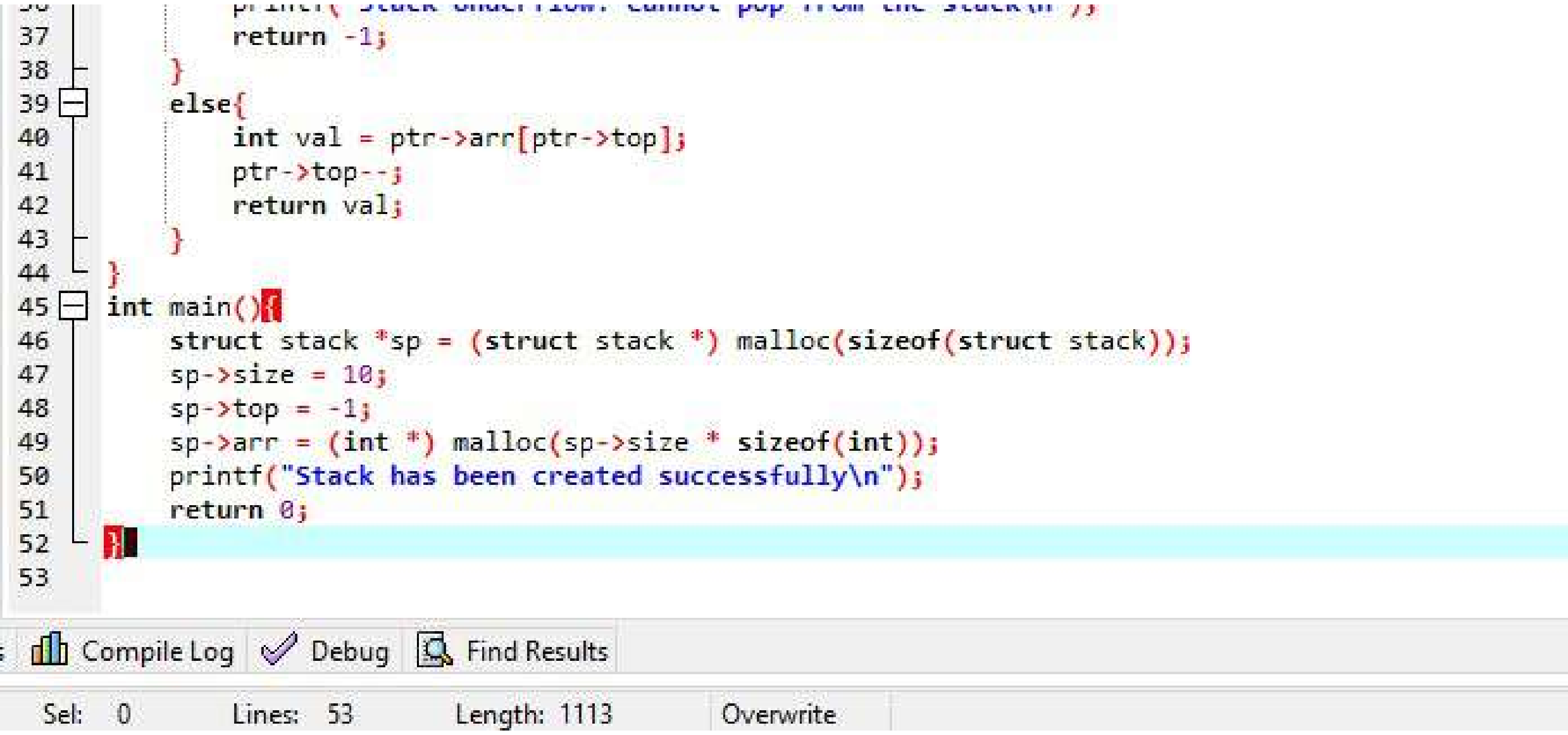
```
STACK POP OPERTIONS
int pop(struct Stack* stack)
{
    if (stack->top == -1)
    {
        printf("Stack Underflow\n");
        return -1;
    int poppedValue = stack->items[stack->top];
    stack->top--;
    printf("%d popped from stack\n", poppedValue);
    return poppedValue;
}

```



```
STACK PEEK OPERATIONS
int peek(struct Stack* stack)
 {
    if (stack->top == -1)
    {
        printf("Stack is empty\n");
        return -1;
    }
    return stack->items[stack->top];
}
```

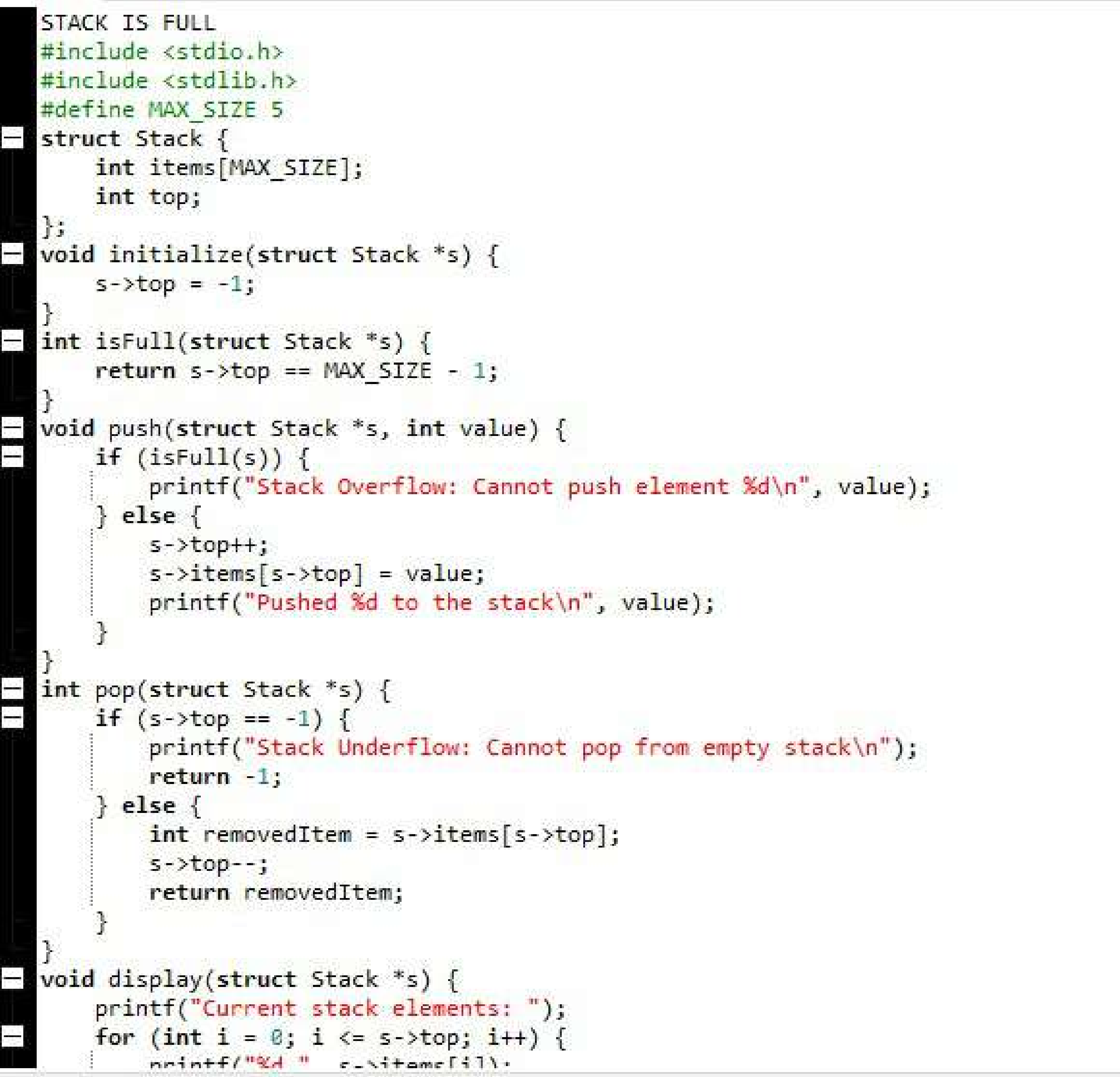


```
STACK ISEMPTY
#include<stdio.h>
#include<stdlib.h>
struct stack{
    int size ;
    int top;
    int * arr;
};
int isEmpty(struct stack* ptr){
    if(ptr->top == -1){
            return 1;
        }
        else{
            return 0;
        }
}
int isFull(struct stack* ptr){
    if(ptr->top == ptr->size - 1){
        return 1;
    }
    else{
        return 0;
    }
}
void push(struct stack* ptr, int val){
    if(isFull(ptr)){
        printf("Stack Overflow! Cannot push %d to the stack\n", val);
    }
    else{
        ptr->top++;
        ptr->arr[ptr->top] = val;
    }
}
int pop(struct stack* ptr){
    if(isEmpty(ptr)){
        printf("Stack Underflow! Cannot pop from the stack\n");
        return -1;
```

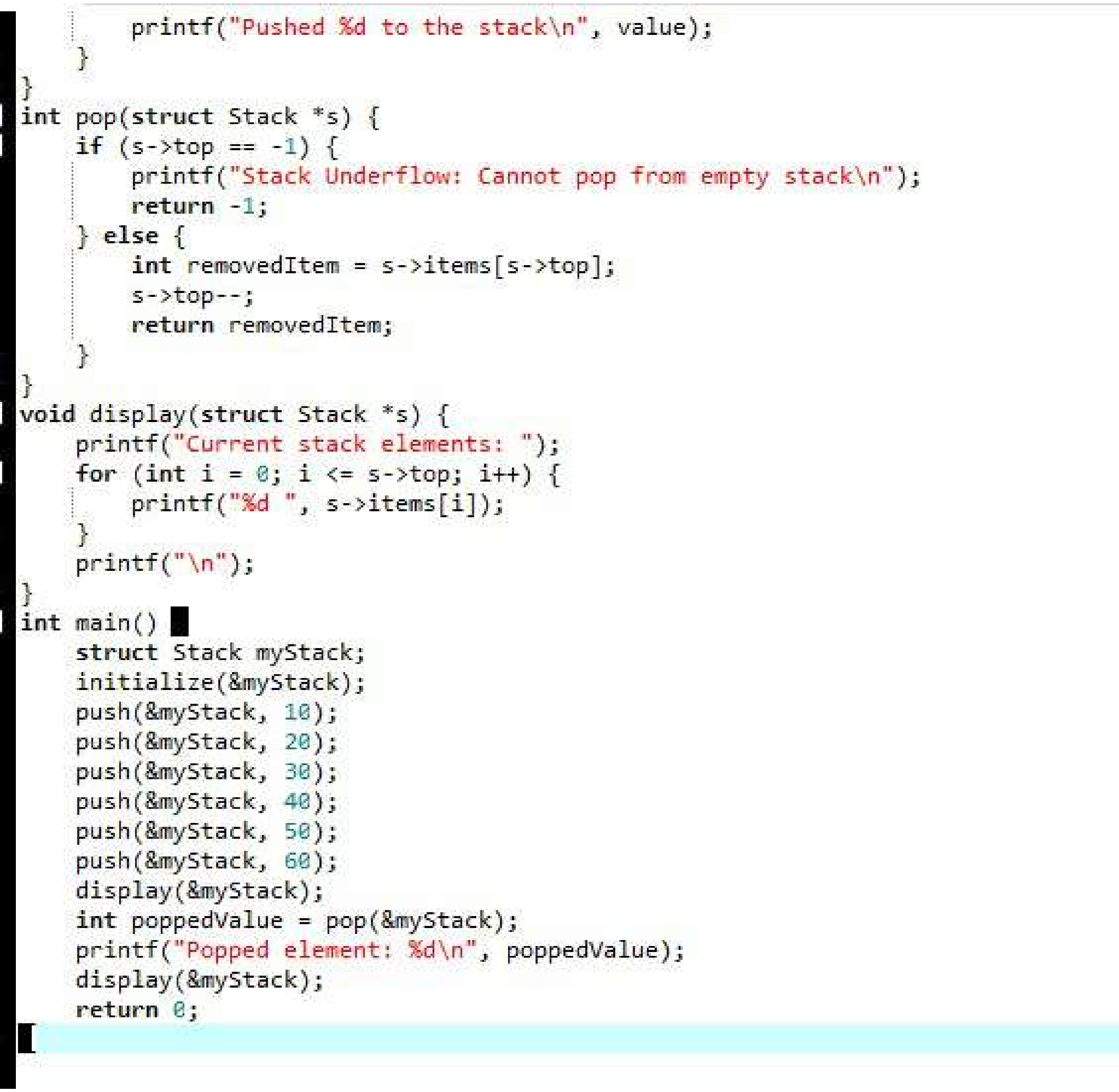


```
        printf("Stack Underflow: cannot pop from the stack\n");
        return -1;
    }
    else{
        int val = ptr->arr[ptr->top];
        ptr->top--;
        return val;
    }
}
int main(){
    struct stack *sp = (struct stack *) malloc(sizeof(struct stack));
    sp->size = 10;
    sp->top = -1;
    sp->arr = (int *) malloc(sp->size * sizeof(int));
    printf("Stack has been created successfully\n");
    return 0;
}
```

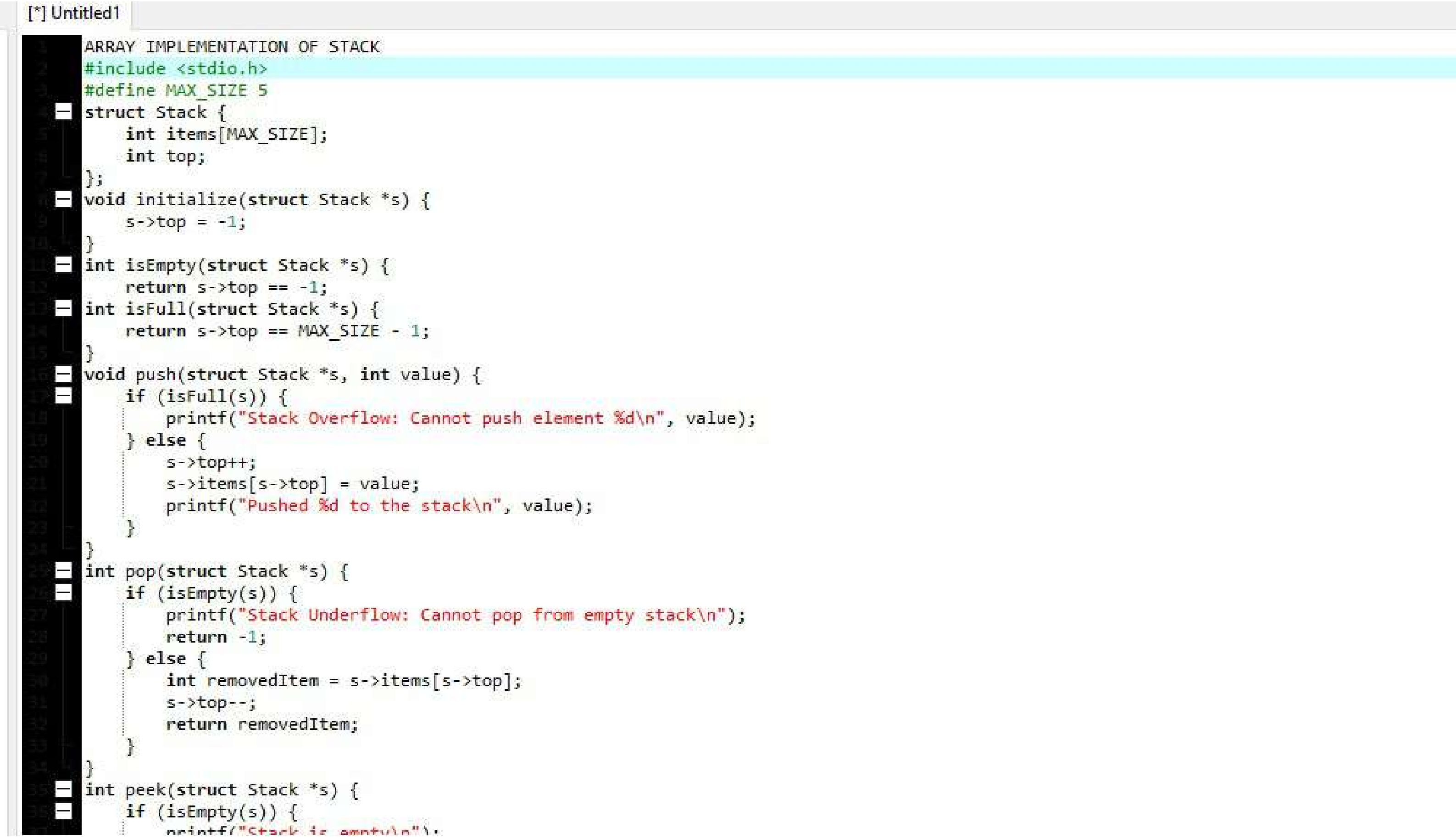
Sel: 0 Lines: 53 Length: 1113 Overwrite



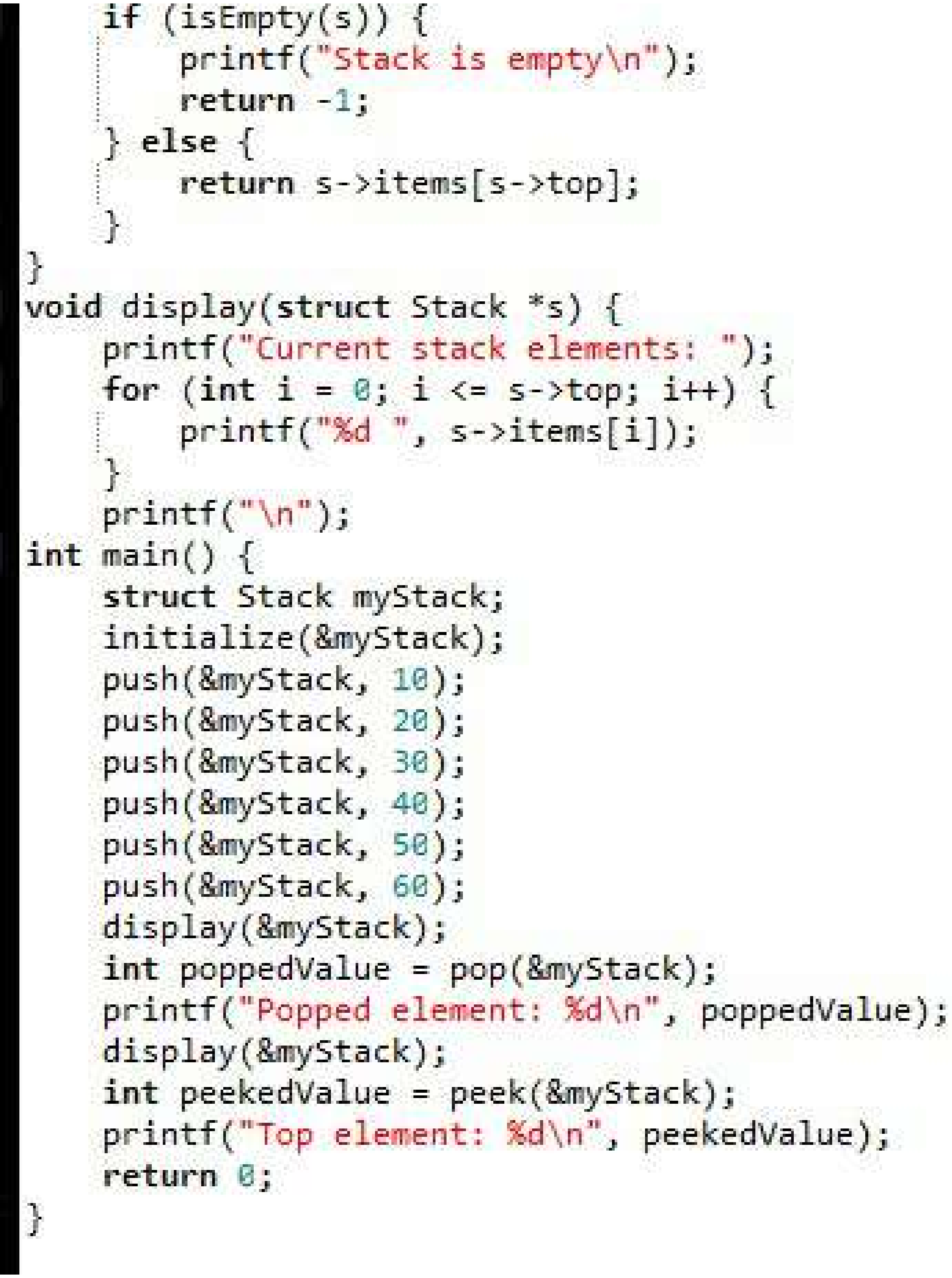
```
STACK IS FULL
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 5
struct Stack {
    int items[MAX_SIZE];
    int top;
};
void initialize(struct Stack *s) {
    s->top = -1;
}
int isFull(struct Stack *s) {
    return s->top == MAX_SIZE - 1;
}
void push(struct Stack *s, int value) {
    if (isFull(s)) {
        printf("Stack Overflow: Cannot push element %d\n", value);
    } else {
        s->top++;
        s->items[s->top] = value;
        printf("Pushed %d to the stack\n", value);
    }
}
int pop(struct Stack *s) {
    if (s->top == -1) {
        printf("Stack Underflow: Cannot pop from empty stack\n");
        return -1;
    } else {
        int removedItem = s->items[s->top];
        s->top--;
        return removedItem;
    }
}
void display(struct Stack *s) {
    printf("Current stack elements: ");
    for (int i = 0; i <= s->top; i++) {
        printf("%d ", s->items[i]);
```



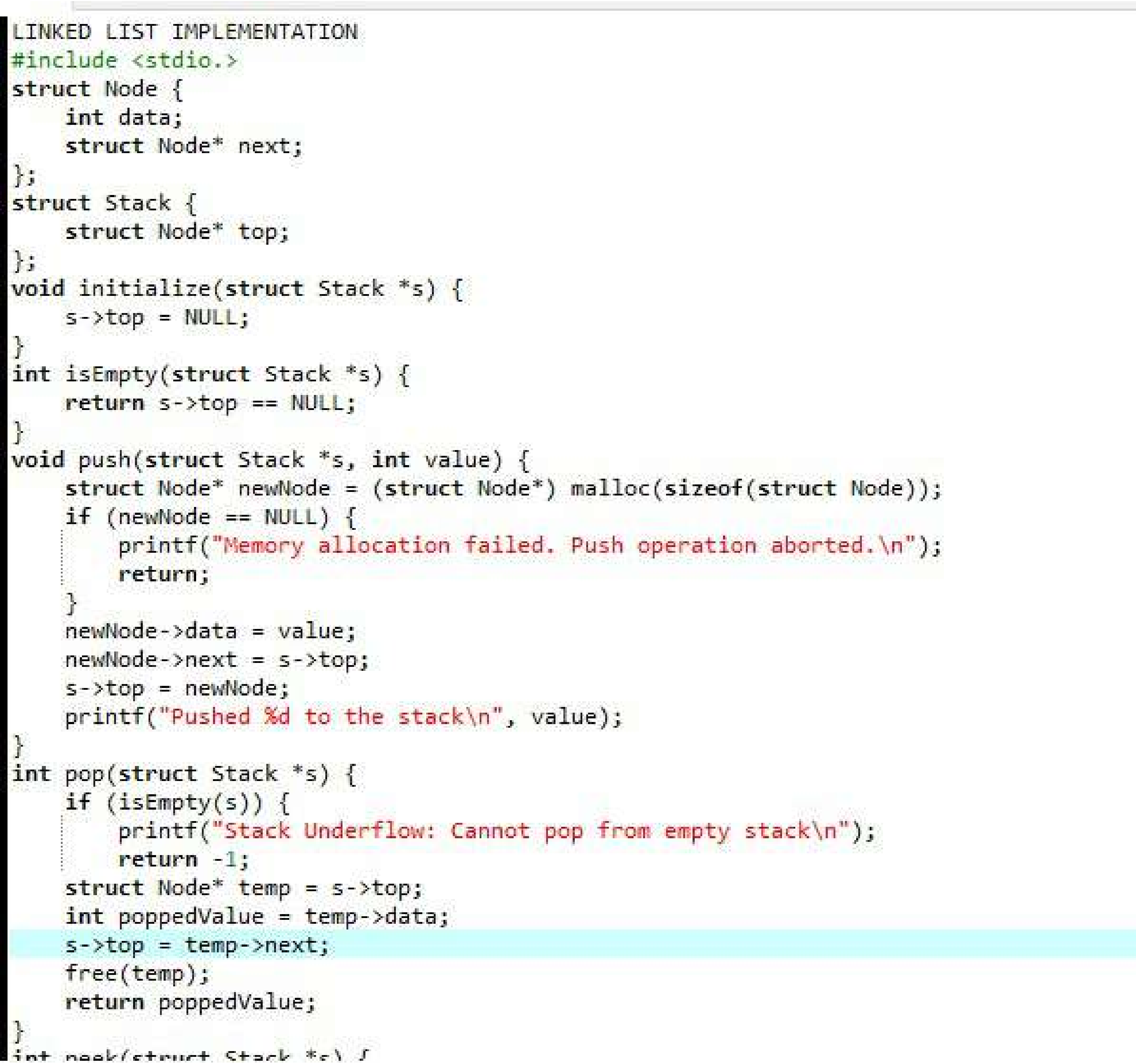
```
        printf("Pushed %d to the stack\n", value);
    }
}
int pop(struct Stack *s) {
    if (s->top == -1) {
        printf("Stack Underflow: Cannot pop from empty stack\n");
        return -1;
    } else {
        int removedItem = s->items[s->top];
        s->top--;
        return removedItem;
    }
}
void display(struct Stack *s) {
    printf("Current stack elements: ");
    for (int i = 0; i <= s->top; i++) {
        printf("%d ", s->items[i]);
    }
    printf("\n");
}
int main() {
    struct Stack myStack;
    initialize(&myStack);
    push(&myStack, 10);
    push(&myStack, 20);
    push(&myStack, 30);
    push(&myStack, 40);
    push(&myStack, 50);
    push(&myStack, 60);
    display(&myStack);
    int poppedValue = pop(&myStack);
    printf("Popped element: %d\n", poppedValue);
    display(&myStack);
    return 0;
}
```



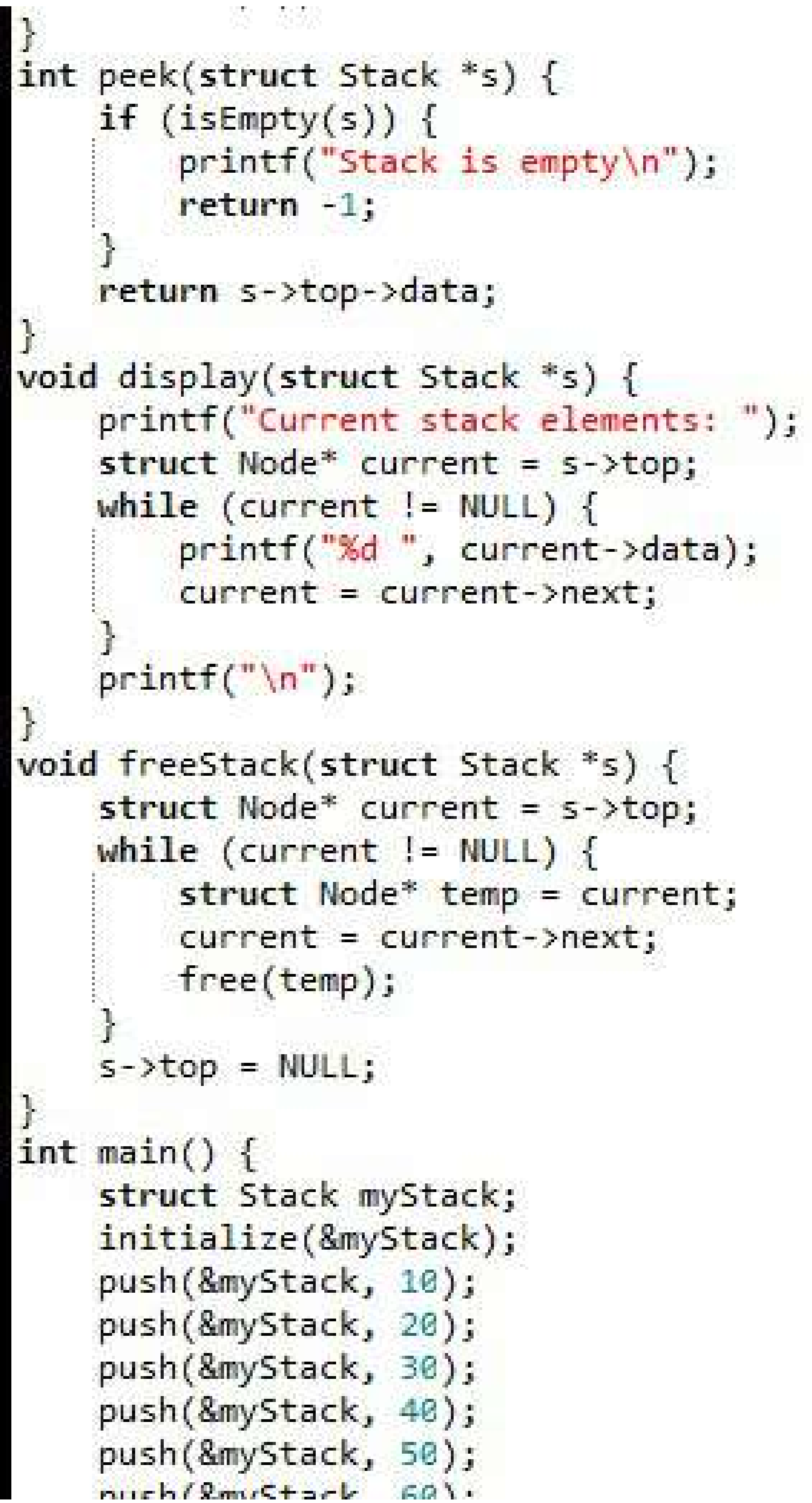
```
ARRAY IMPLEMENTATION OF STACK
#include <stdio.h>
#define MAX_SIZE 5
struct Stack {
    int items[MAX_SIZE];
    int top;
};
void initialize(struct Stack *s) {
    s->top = -1;
}
int isEmpty(struct Stack *s) {
    return s->top == -1;
int isFull(struct Stack *s) {
    return s->top == MAX_SIZE - 1;
}
void push(struct Stack *s, int value) {
    if (isFull(s)) {
        printf("Stack Overflow: Cannot push element %d\n", value);
    } else {
        s->top++;
        s->items[s->top] = value;
        printf("Pushed %d to the stack\n", value);
    }
}
int pop(struct Stack *s) {
    if (isEmpty(s)) {
        printf("Stack Underflow: Cannot pop from empty stack\n");
        return -1;
    } else {
        int removedItem = s->items[s->top];
        s->top--;
        return removedItem;
    }
}
int peek(struct Stack *s) {
    if (isEmpty(s)) {
        printf("Stack is empty\n");
```



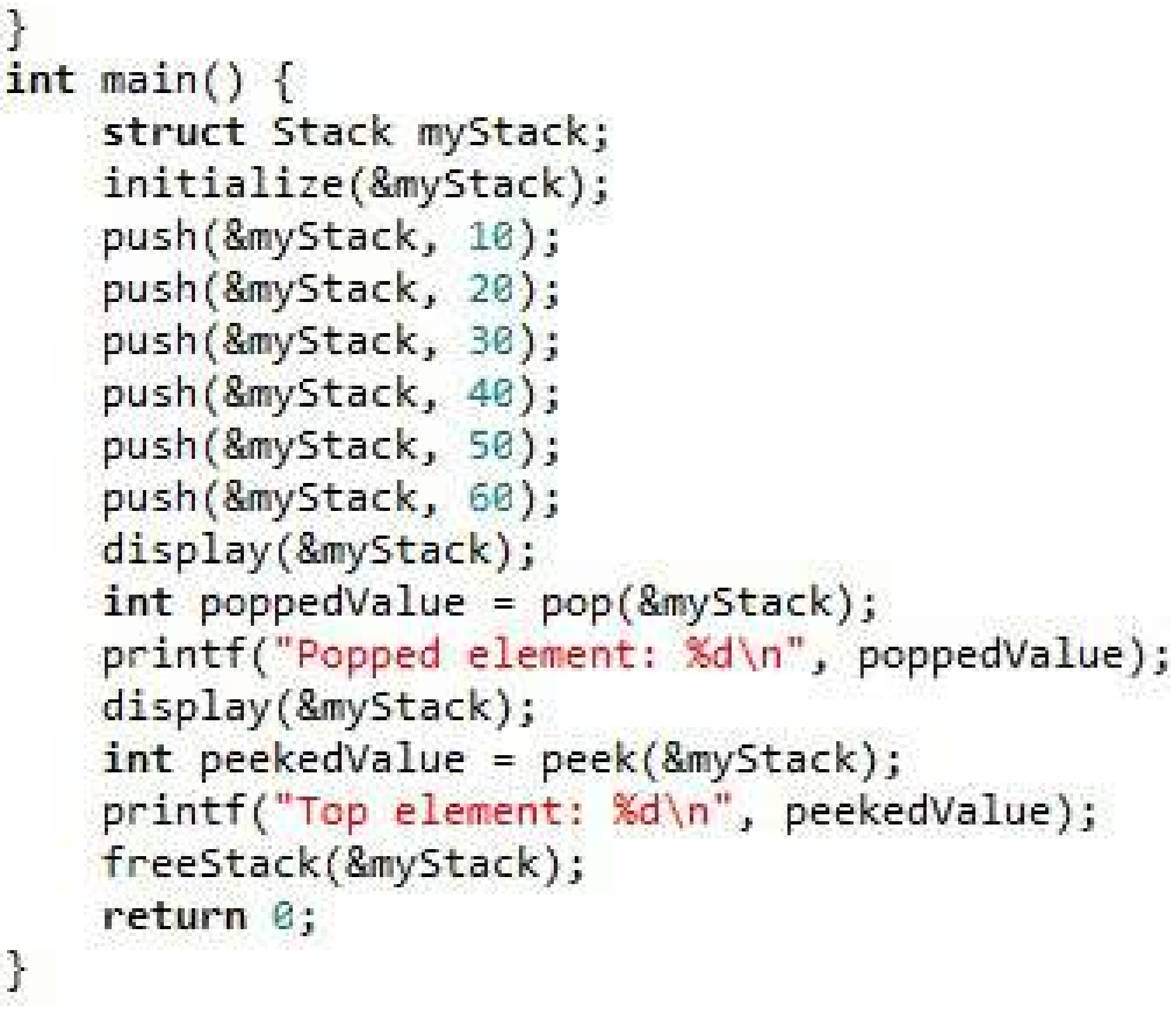
```
    if (isEmpty(s)) {
        printf("Stack is empty\n");
        return -1;
    } else {
        return s->items[s->top];
    }
}
void display(struct Stack *s) {
    printf("Current stack elements: ");
    for (int i = 0; i <= s->top; i++) {
        printf("%d ", s->items[i]);
    }
    printf("\n");
int main() {
    struct Stack myStack;
    initialize(&myStack);
    push(&myStack, 10);
    push(&myStack, 20);
    push(&myStack, 30);
    push(&myStack, 40);
    push(&myStack, 50);
    push(&myStack, 60);
    display(&myStack);
    int poppedValue = pop(&myStack);
    printf("Popped element: %d\n", poppedValue);
    display(&myStack);
    int peekedValue = peek(&myStack);
    printf("Top element: %d\n", peekedValue);
    return 0;
}
```



```
LINKED LIST IMPLEMENTATION
#include <stdio.>
struct Node {
    int data;
    struct Node* next;
};
struct Stack {
    struct Node* top;
};
void initialize(struct Stack *s) {
    s->top = NULL;
}
int isEmpty(struct Stack *s) {
    return s->top == NULL;
}
void push(struct Stack *s, int value) {
    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed. Push operation aborted.\n");
        return;
    }
    newNode->data = value;
    newNode->next = s->top;
    s->top = newNode;
    printf("Pushed %d to the stack\n", value);
}
int pop(struct Stack *s) {
    if (isEmpty(s)) {
        printf("Stack Underflow: Cannot pop from empty stack\n");
        return -1;
    struct Node* temp = s->top;
    int poppedValue = temp->data;
    s->top = temp->next;
    free(temp);
    return poppedValue;
}
int peek(struct Stack *s) {
```



```
}
int peek(struct Stack *s) {
    if (isEmpty(s)) {
        printf("Stack is empty\n");
        return -1;
    }
    return s->top->data;
}
void display(struct Stack *s) {
    printf("Current stack elements: ");
    struct Node* current = s->top;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}
void freeStack(struct Stack *s) {
    struct Node* current = s->top;
    while (current != NULL) {
        struct Node* temp = current;
        current = current->next;
        free(temp);
    }
    s->top = NULL;
}
int main() {
    struct Stack myStack;
    initialize(&myStack);
    push(&myStack, 10);
    push(&myStack, 20);
    push(&myStack, 30);
    push(&myStack, 40);
    push(&myStack, 50);
    push(&myStack, 60);
```



```
}
int main() {
    struct Stack myStack;
    initialize(&myStack);
    push(&myStack, 10);
    push(&myStack, 20);
    push(&myStack, 30);
    push(&myStack, 40);
    push(&myStack, 50);
    push(&myStack, 60);
    display(&myStack);
    int poppedValue = pop(&myStack);
    printf("Popped element: %d\n", poppedValue);
    display(&myStack);
    int peekedValue = peek(&myStack);
    printf("Top element: %d\n", peekedValue);
    freeStack(&myStack);
    return 0;
}
```