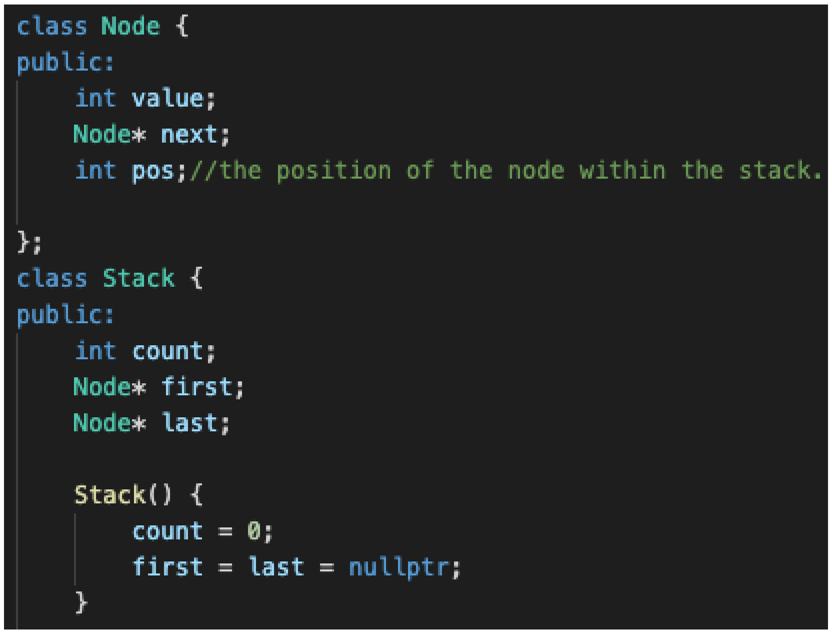
Problem 4 - By Hugo Barahona

My own stack, displaying content and changing a value.



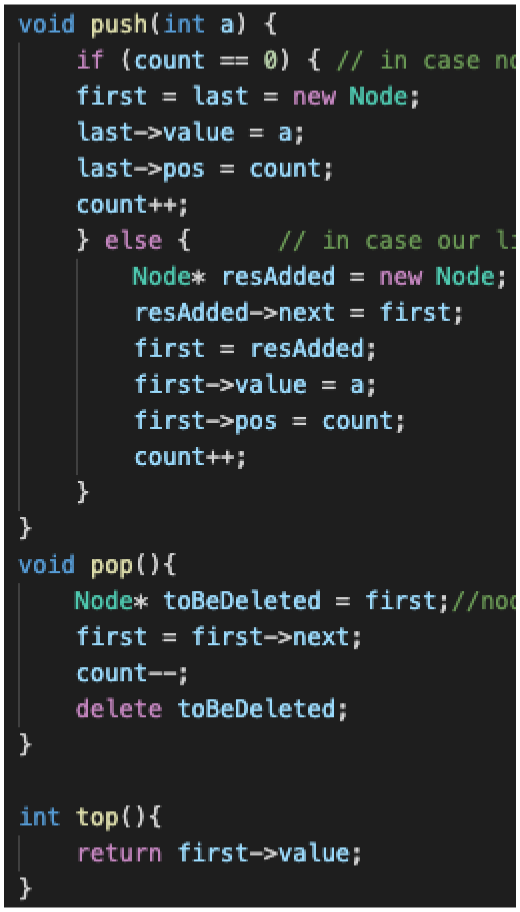
```
class Node {
public:
    int value;
    Node* next;
    int pos;//the position of the node within the stack.

};
class Stack {
public:
    int count;
    Node* first;
    Node* last;

    Stack() {
        count = 0;
        first = last = nullptr;
    }
```

I make use of two classes, class Node and class Stack. For the class Node, each object contains two int variables, one for a value, and one for its position (to be later used in the Stack), and a pointer to a Node called next.

In the Stack class, I made it using a linked list but built backwards. The stack keeps a count of the elements within, and the count variable is also used to set the position of the nodes, two pointers, one of the first Node, and one for the Last node.



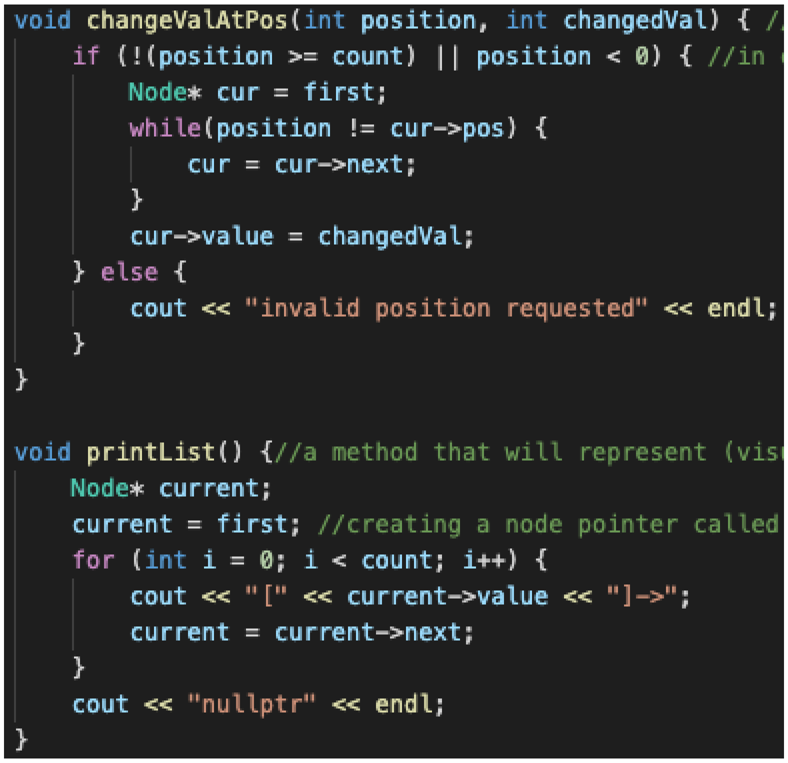
```
void push(int a) {
    if (count == 0) { // in case no
    first = last = new Node;
    last->value = a;
    last->pos = count;
    count++;
    } else {      // in case our li
        Node* resAdded = new Node;
        resAdded->next = first;
        first = resAdded;
        first->value = a;
        first->pos = count;
        count++;
    }
}
void pop(){
    Node* toBeDeleted = first;//noc
    first = first->next;
    count--;
    delete toBeDeleted;
}

int top(){
    return first->value;
}
```

Within the Stack class, we have a push function that is in charge of building the stack backwards. In other words, every new element becomes the element that points to the previous elements. The “first” pointer is the top of the stack and the head of the list.

A pop function also exists, and it ensures that every node that is popped is properly deleted and its memory is deallocated.

A top function that just returns the “first” pointer. Which is the last element added. While writing this, I just noticed that I should have named it last, instead of first.



```
void changeValAtPos(int position, int changedVal) { /
    if (!(position >= count) || position < 0) { //in
        Node* cur = first;
        while(position != cur->pos) {
            cur = cur->next;
        }
        cur->value = changedVal;
    } else {
        cout << "invalid position requested" << endl;
    }
}

void printList() {//a method that will represent (vis
    Node* current;
    current = first; //creating a node pointer called
    for (int i = 0; i < count; i++) {
        cout << "[" << current->value << "]->";
        current = current->next;
    }
    cout << "nullptr" << endl;
}
```

Finally, a function that changes the value of an specified position, if the position is more or equal than the count of nodes within the stack, or if the position is less than zero, then the function prints a message. Otherwise, it changes the value of that position. Parameters are the position and the value to be changed.

A printList function that works the same way as it would in a regular list.