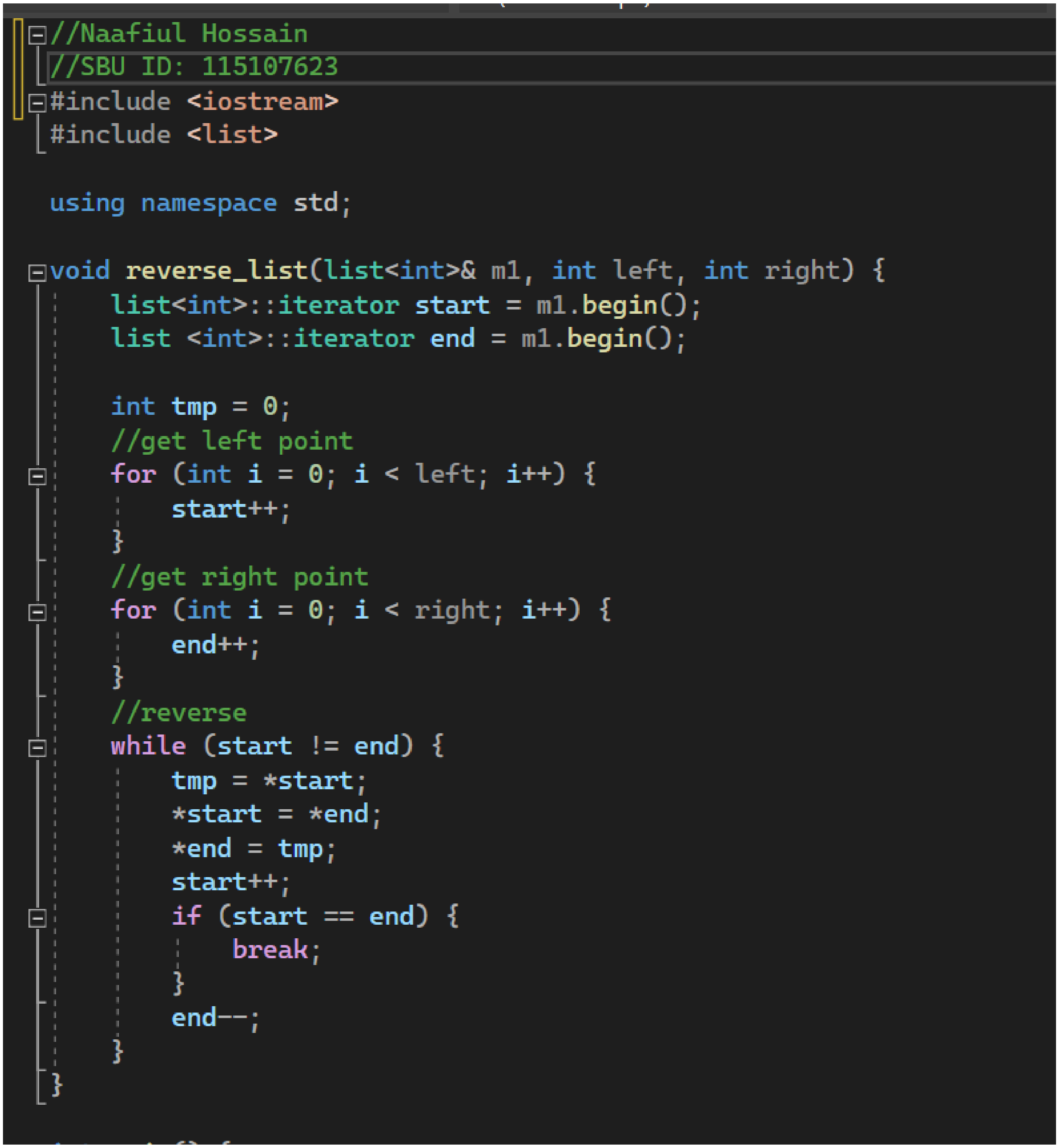
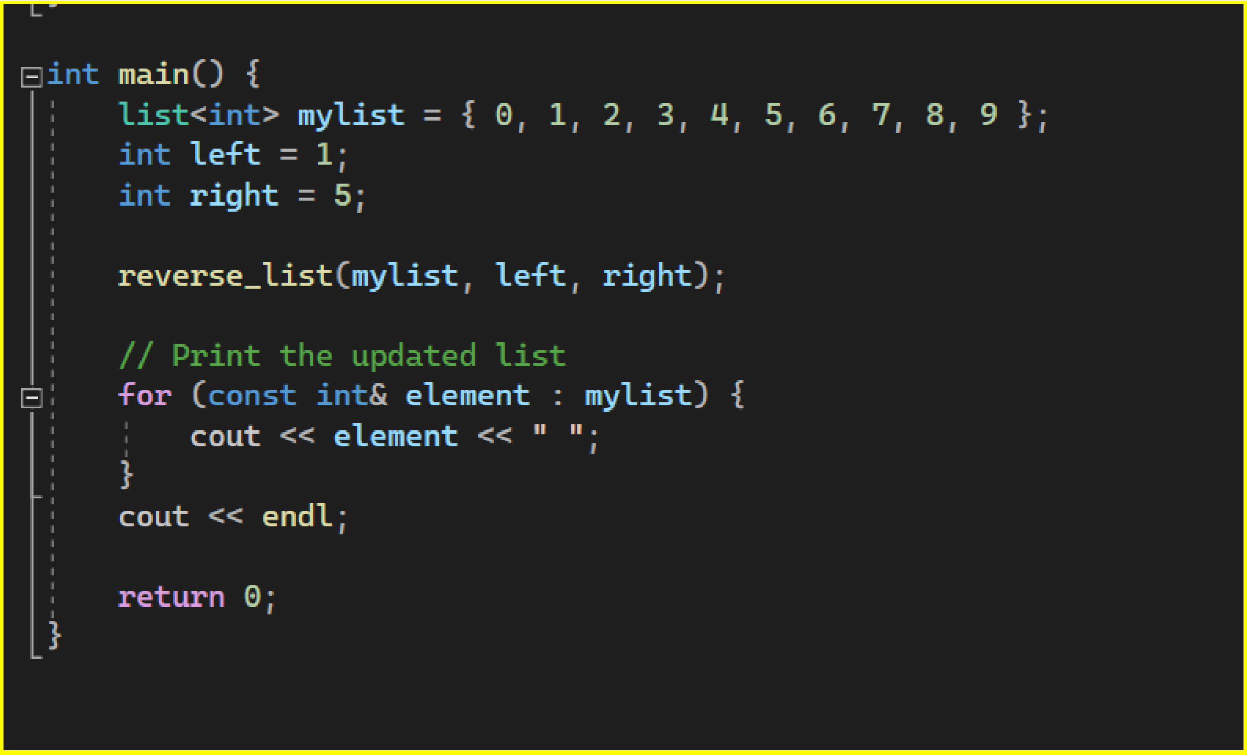
Naafiul Hossain
ESE224
115107623
Tuesday 10-12:50am

Problem 1:

Main.cpp

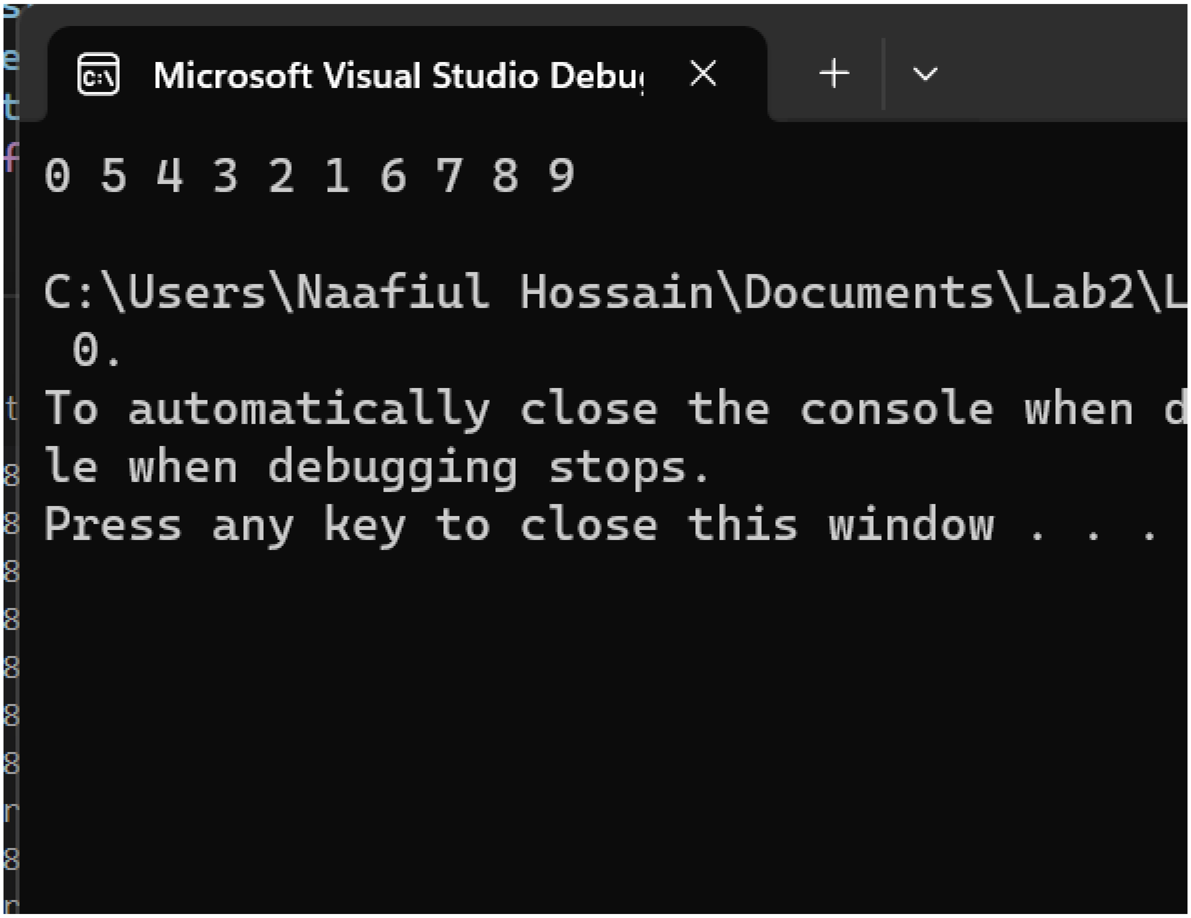


```cpp
//Naafiul Hossain
//SBU ID: 115107623
#include <iostream>
#include <list>

using namespace std;

void reverse_list(list<int>& m1, int left, int right) {
    list<int>::iterator start = m1.begin();
    list <int>::iterator end = m1.begin();

    int tmp = 0;
    //get left point
    for (int i = 0; i < left; i++) {
        start++;
    }
    //get right point
    for (int i = 0; i < right; i++) {
        end++;
    }
    //reverse
    while (start != end) {
        tmp = *start;
        *start = *end;
        *end = tmp;
        start++;
        if (start == end) {
            break;
        }
        end--;
    }
}
```



```
int main() {
    list<int> mylist = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    int left = 1;
    int right = 5;

    reverse_list(mylist, left, right);

    // Print the updated list
    for (const int& element : mylist) {
        cout << element << " ";
    }
    cout << endl;

    return 0;
}
```

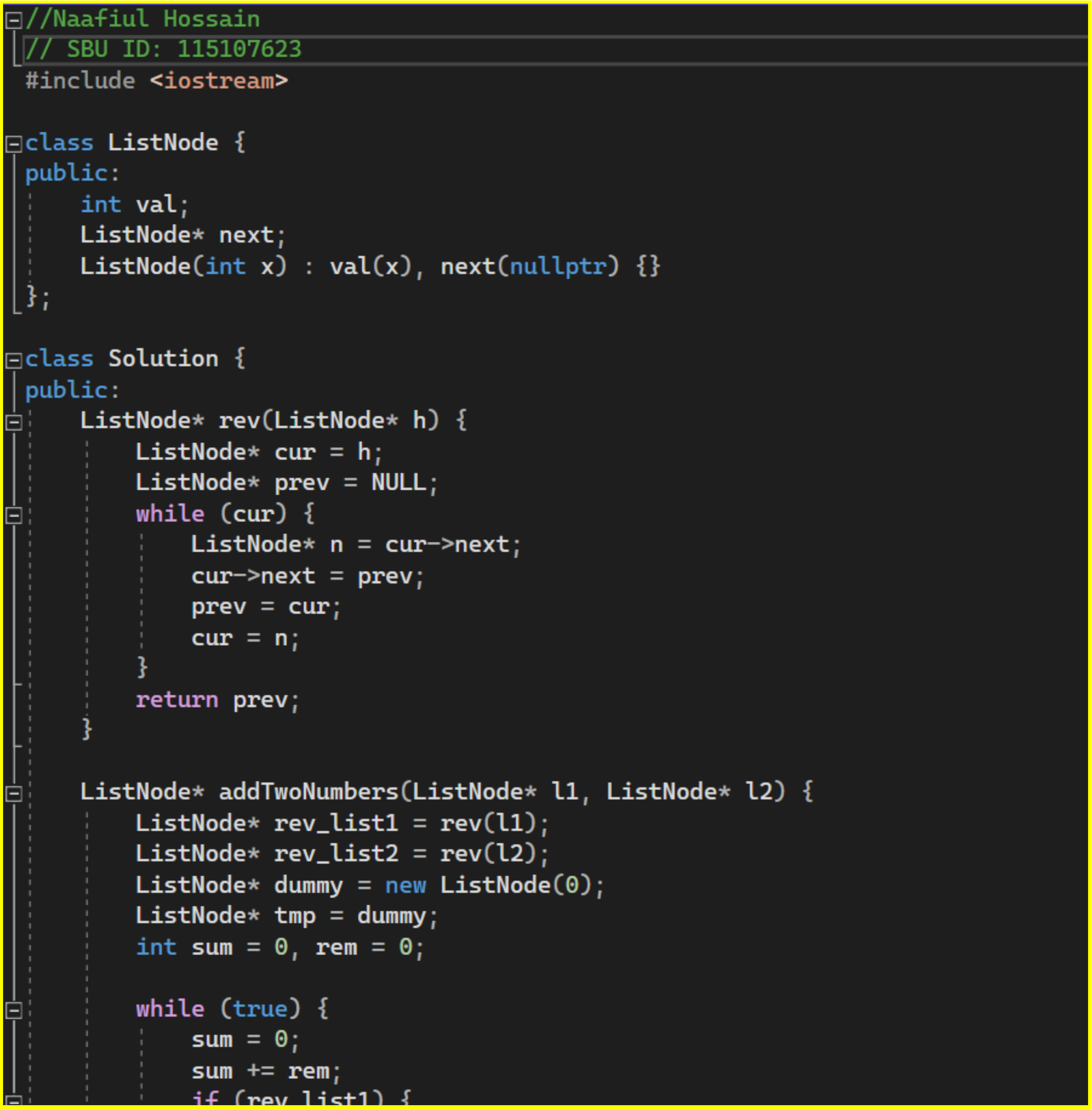
Screenshot of the running program:



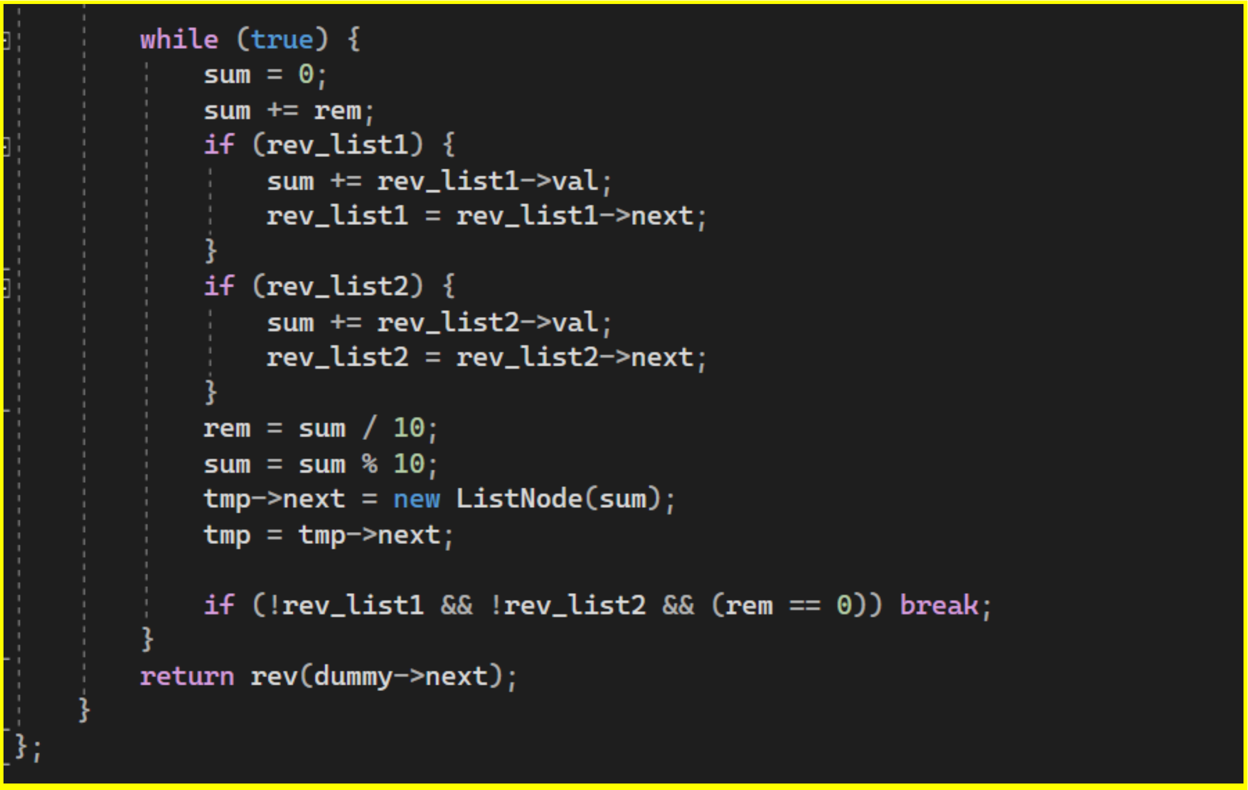
```
0 5 4 3 2 1 6 7 8 9

C:\Users\Naafiul Hossain\Documents\Lab2\L
 0.
To automatically close the console when d
le when debugging stops.
Press any key to close this window . . .
```

Problem 2

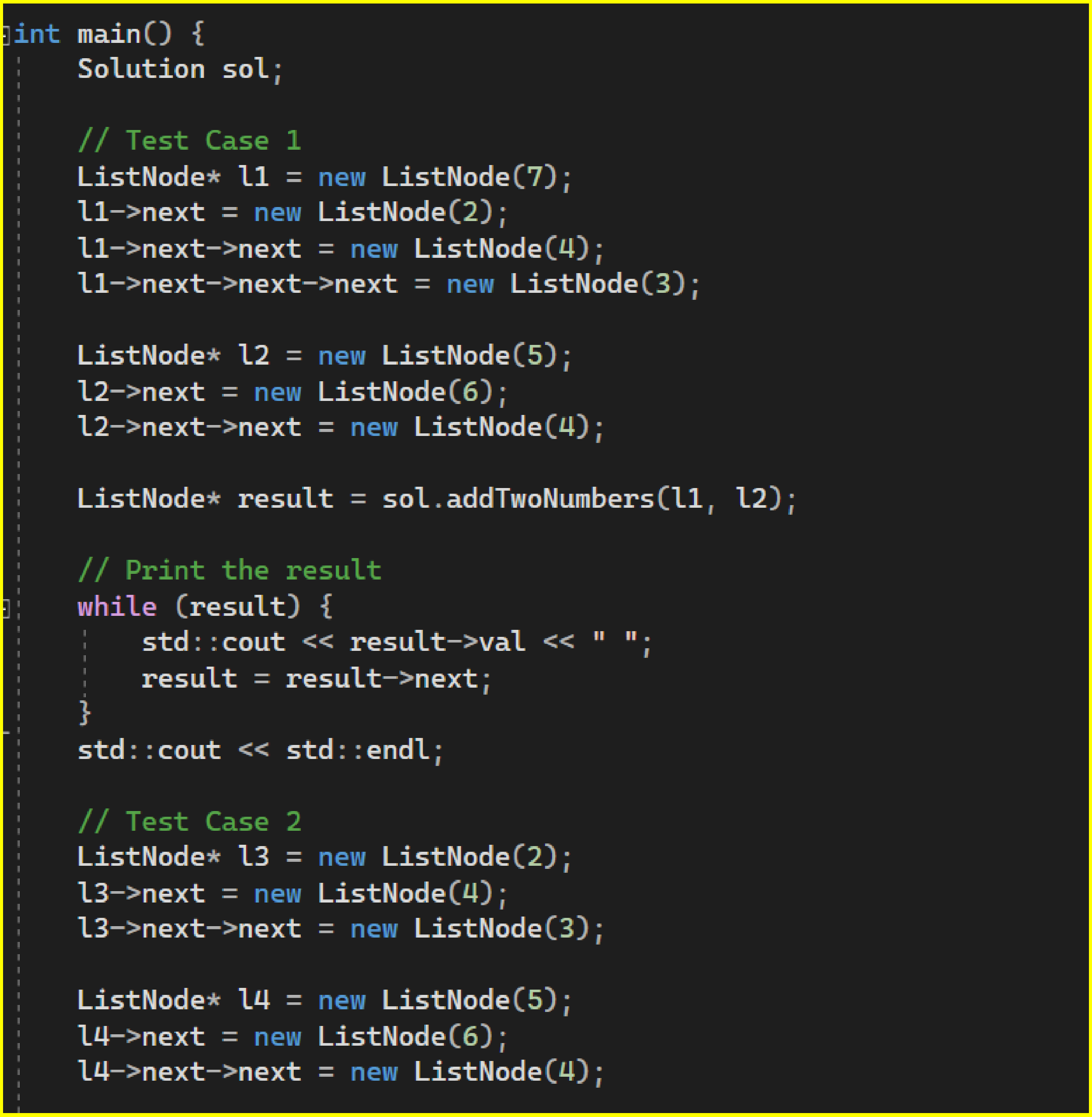
main.h



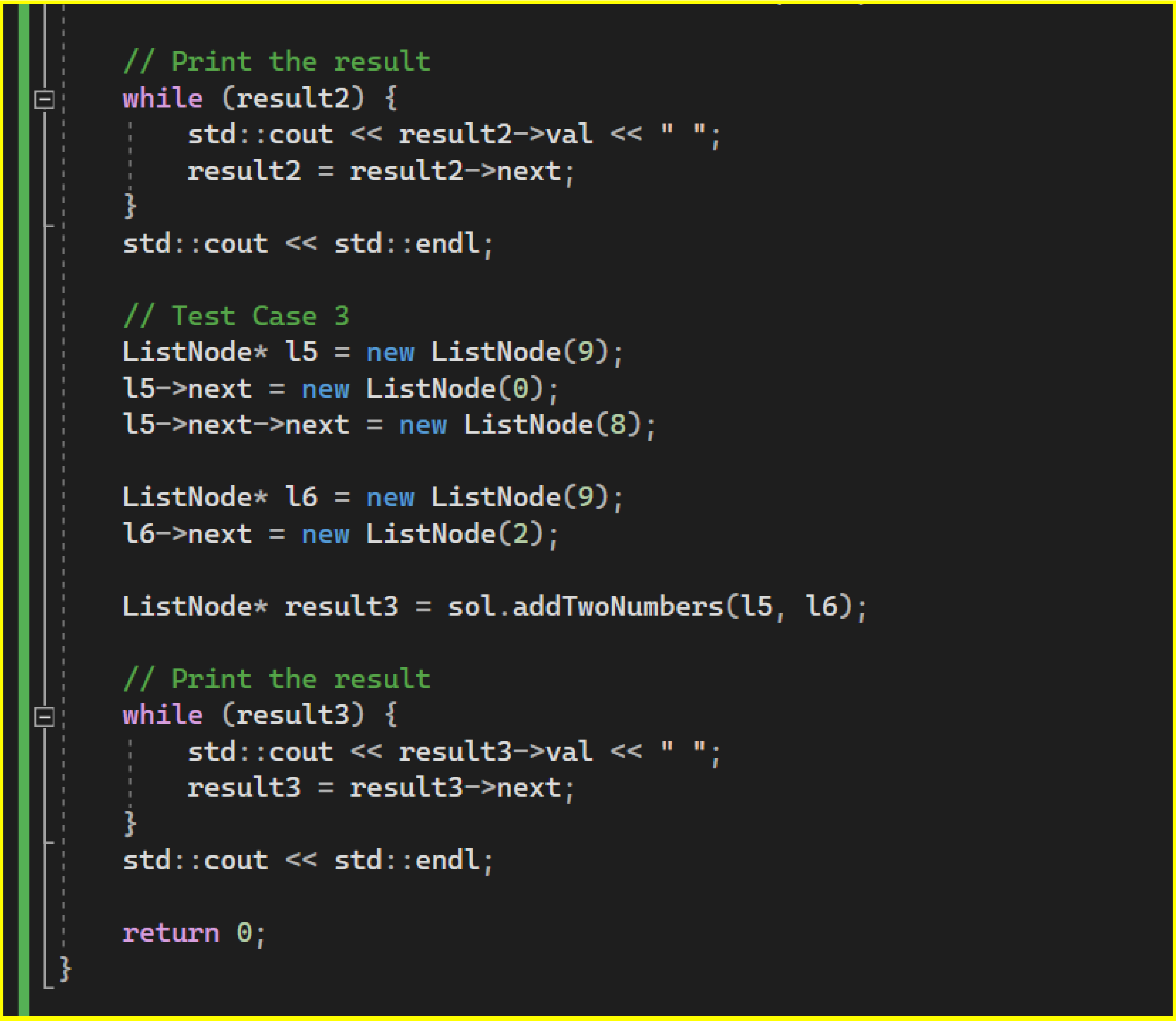
```
//Naafiul Hossain
// SBU ID: 115107623
#include <iostream>

class ListNode {
public:
    int val;
    ListNode* next;
    ListNode(int x) : val(x), next(nullptr) {}
};

class Solution {
public:
    ListNode* rev(ListNode* h) {
        ListNode* cur = h;
        ListNode* prev = NULL;
        while (cur) {
            ListNode* n = cur->next;
            cur->next = prev;
            prev = cur;
            cur = n;
        }
        return prev;
    }

    ListNode* addTwoNumbers(ListNode* l1, ListNode* l2) {
        ListNode* rev_list1 = rev(l1);
        ListNode* rev_list2 = rev(l2);
        ListNode* dummy = new ListNode(0);
        ListNode* tmp = dummy;
        int sum = 0, rem = 0;

        while (true) {
            sum = 0;
            sum += rem;
            if (rev_list1) {
```



```
        while (true) {
            sum = 0;
            sum += rem;
            if (rev_list1) {
                sum += rev_list1->val;
                rev_list1 = rev_list1->next;
            }
            if (rev_list2) {
                sum += rev_list2->val;
                rev_list2 = rev_list2->next;
            }
            rem = sum / 10;
            sum = sum % 10;
            tmp->next = new ListNode(sum);
            tmp = tmp->next;

            if (!rev_list1 && !rev_list2 && (rem == 0)) break;
        }
        return rev(dummy->next);
    }
};
```

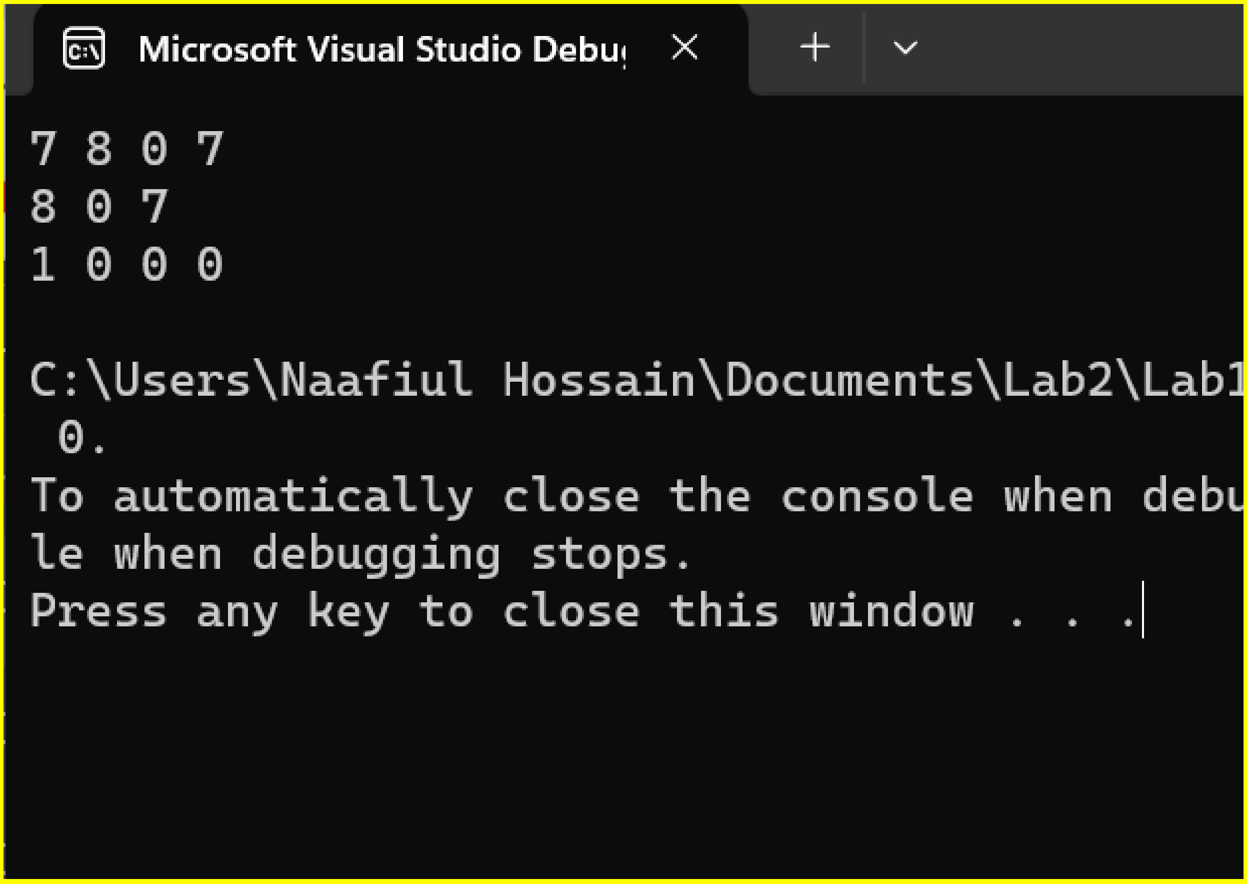


```
int main() {
    Solution sol;

    // Test Case 1
    ListNode* l1 = new ListNode(7);
    l1->next = new ListNode(2);
    l1->next->next = new ListNode(4);
    l1->next->next->next = new ListNode(3);

    ListNode* l2 = new ListNode(5);
    l2->next = new ListNode(6);
    l2->next->next = new ListNode(4);

    ListNode* result = sol.addTwoNumbers(l1, l2);

    // Print the result
    while (result) {
        std::cout << result->val << " ";
        result = result->next;
    }
    std::cout << std::endl;

    // Test Case 2
    ListNode* l3 = new ListNode(2);
    l3->next = new ListNode(4);
    l3->next->next = new ListNode(3);

    ListNode* l4 = new ListNode(5);
    l4->next = new ListNode(6);
    l4->next->next = new ListNode(4);
```



```
    // Print the result
    while (result2) {
        std::cout << result2->val << " ";
        result2 = result2->next;
    }
    std::cout << std::endl;

    // Test Case 3
    ListNode* l5 = new ListNode(9);
    l5->next = new ListNode(0);
    l5->next->next = new ListNode(8);

    ListNode* l6 = new ListNode(9);
    l6->next = new ListNode(2);

    ListNode* result3 = sol.addTwoNumbers(l5, l6);

    // Print the result
    while (result3) {
        std::cout << result3->val << " ";
        result3 = result3->next;
    }
    std::cout << std::endl;

    return 0;
}
```

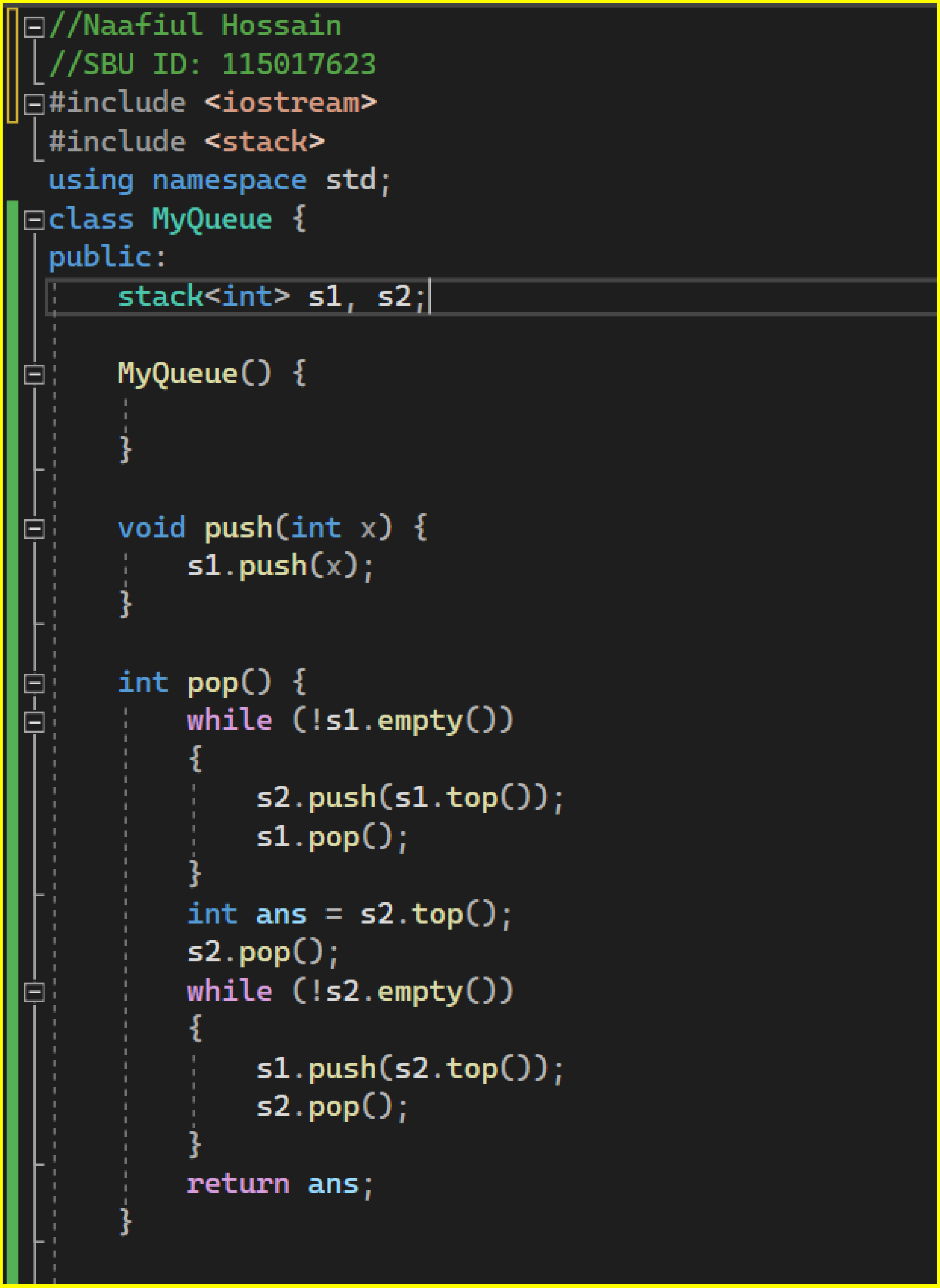
Running of the Program:



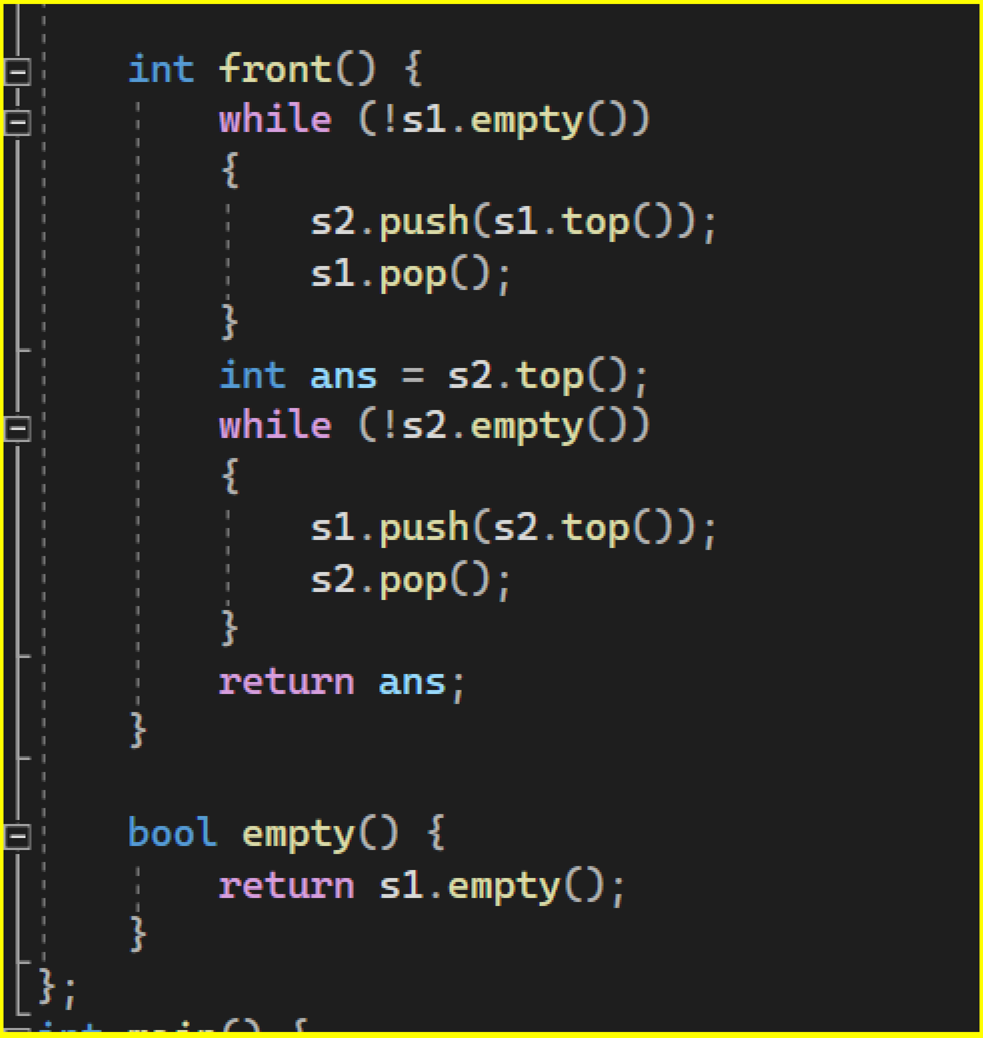
```
7 8 0 7
8 0 7
1 0 0 0

C:\Users\Naafiul Hossain\Documents\Lab2\Lab1
 0.
To automatically close the console when debu
le when debugging stops.
Press any key to close this window . . .
```

Problem 3

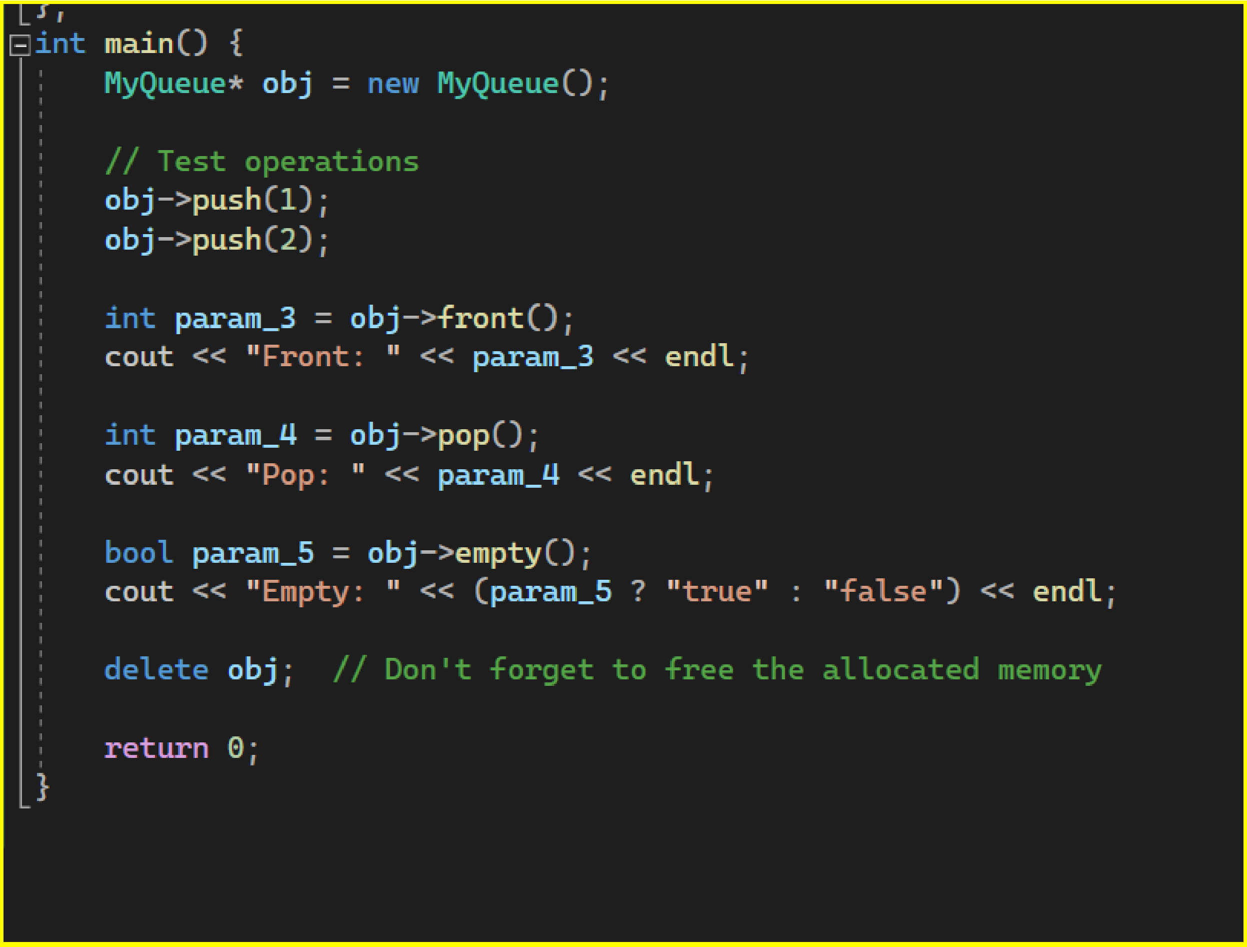
Main.cpp



```cpp
//Naafiul Hossain
//SBU ID: 115017623
#include <iostream>
#include <stack>
using namespace std;
class MyQueue {
public:
    stack<int> s1, s2;

    MyQueue() {

    }

    void push(int x) {
        s1.push(x);
    }

    int pop() {
        while (!s1.empty())
        {
            s2.push(s1.top());
            s1.pop();
        }
        int ans = s2.top();
        s2.pop();
        while (!s2.empty())
        {
            s1.push(s2.top());
            s2.pop();
        }
        return ans;
    }
```

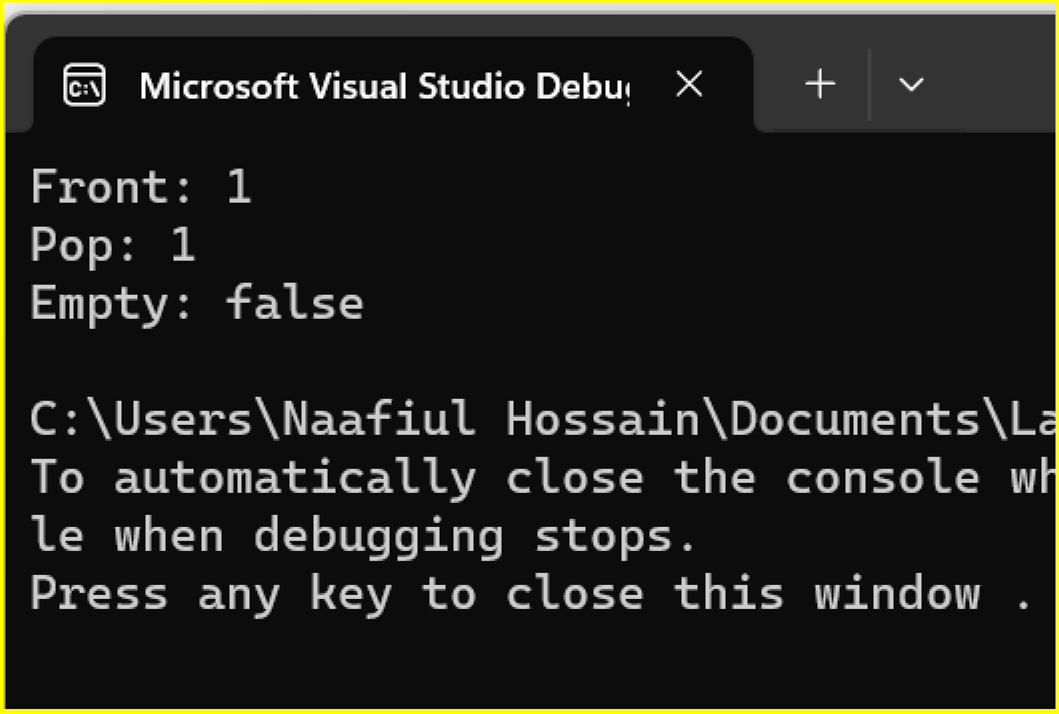


```
    int front() {
        while (!s1.empty())
        {
            s2.push(s1.top());
            s1.pop();
        }
        int ans = s2.top();
        while (!s2.empty())
        {
            s1.push(s2.top());
            s2.pop();
        }
        return ans;
    }

    bool empty() {
        return s1.empty();
    }
};
int main() {
```



```
};
int main() {
    MyQueue* obj = new MyQueue();

    // Test operations
    obj->push(1);
    obj->push(2);

    int param_3 = obj->front();
    cout << "Front: " << param_3 << endl;

    int param_4 = obj->pop();
    cout << "Pop: " << param_4 << endl;

    bool param_5 = obj->empty();
    cout << "Empty: " << (param_5 ? "true" : "false") << endl;

    delete obj;  // Don't forget to free the allocated memory

    return 0;
}
```

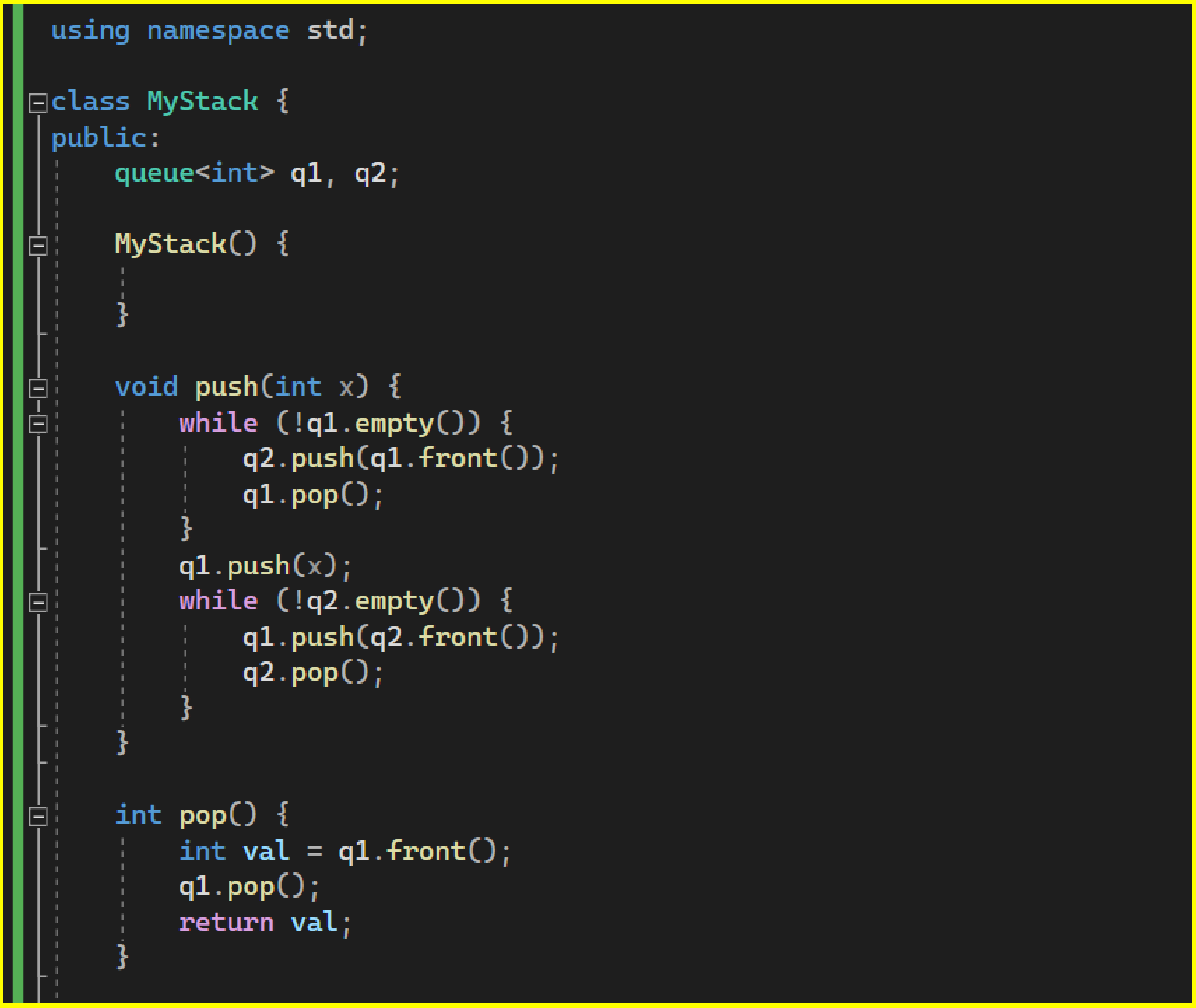
Screenshot of the running program:



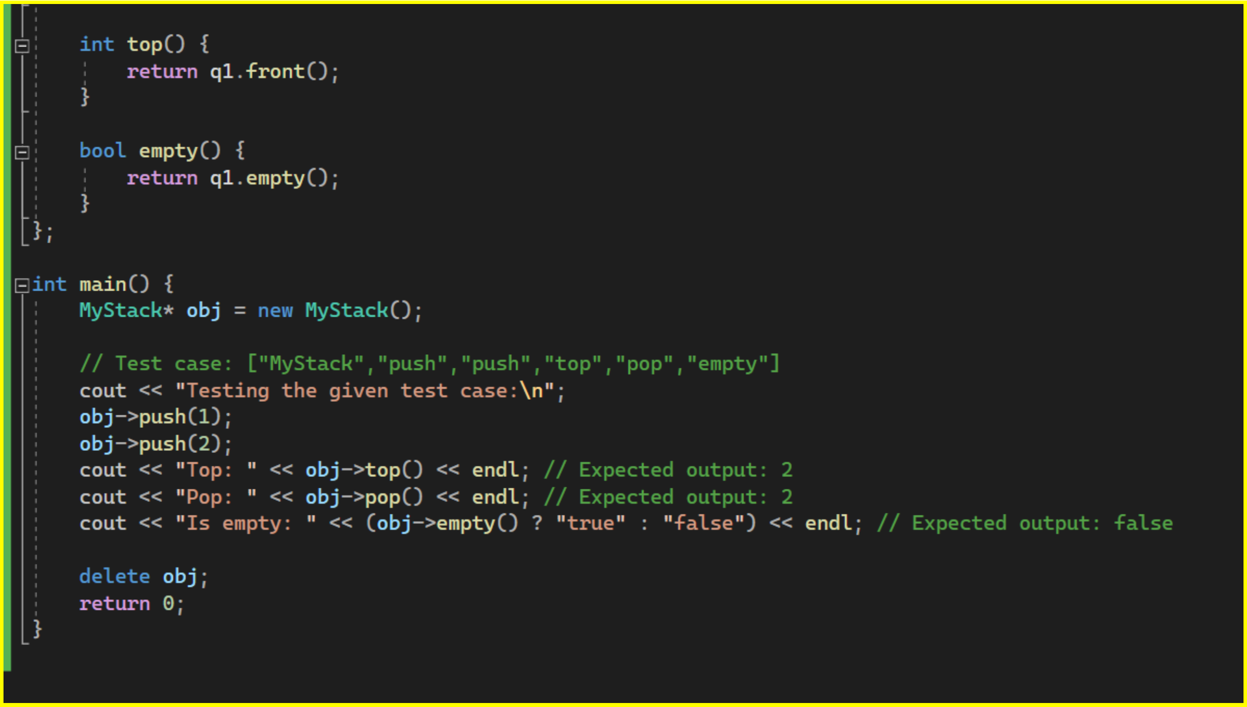
```
Front: 1
Pop: 1
Empty: false

C:\Users\Naafiul Hossain\Documents\La
To automatically close the console wh
le when debugging stops.
Press any key to close this window .
```

Problem 4

Main.cpp

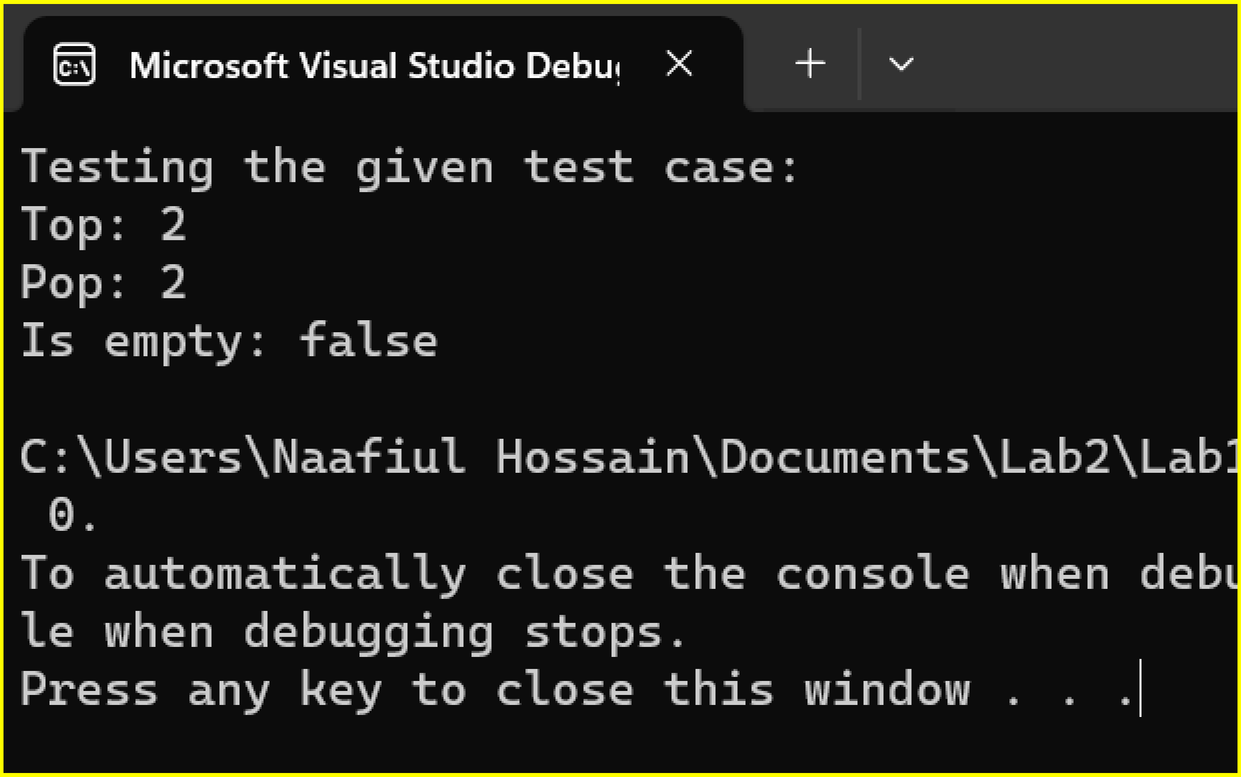


```
using namespace std;

class MyStack {
public:
    queue<int> q1, q2;

    MyStack() {

    }

    void push(int x) {
        while (!q1.empty()) {
            q2.push(q1.front());
            q1.pop();
        }
        q1.push(x);
        while (!q2.empty()) {
            q1.push(q2.front());
            q2.pop();
        }
    }

    int pop() {
        int val = q1.front();
        q1.pop();
        return val;
    }
```



```
    int top() {
        return q1.front();
    }

    bool empty() {
        return q1.empty();
    }
};

int main() {
    MyStack* obj = new MyStack();

    // Test case: ["MyStack","push","push","top","pop","empty"]
    cout << "Testing the given test case:\n";
    obj->push(1);
    obj->push(2);
    cout << "Top: " << obj->top() << endl; // Expected output: 2
    cout << "Pop: " << obj->pop() << endl; // Expected output: 2
    cout << "Is empty: " << (obj->empty() ? "true" : "false") << endl; // Expected output: false

    delete obj;
    return 0;
}
```

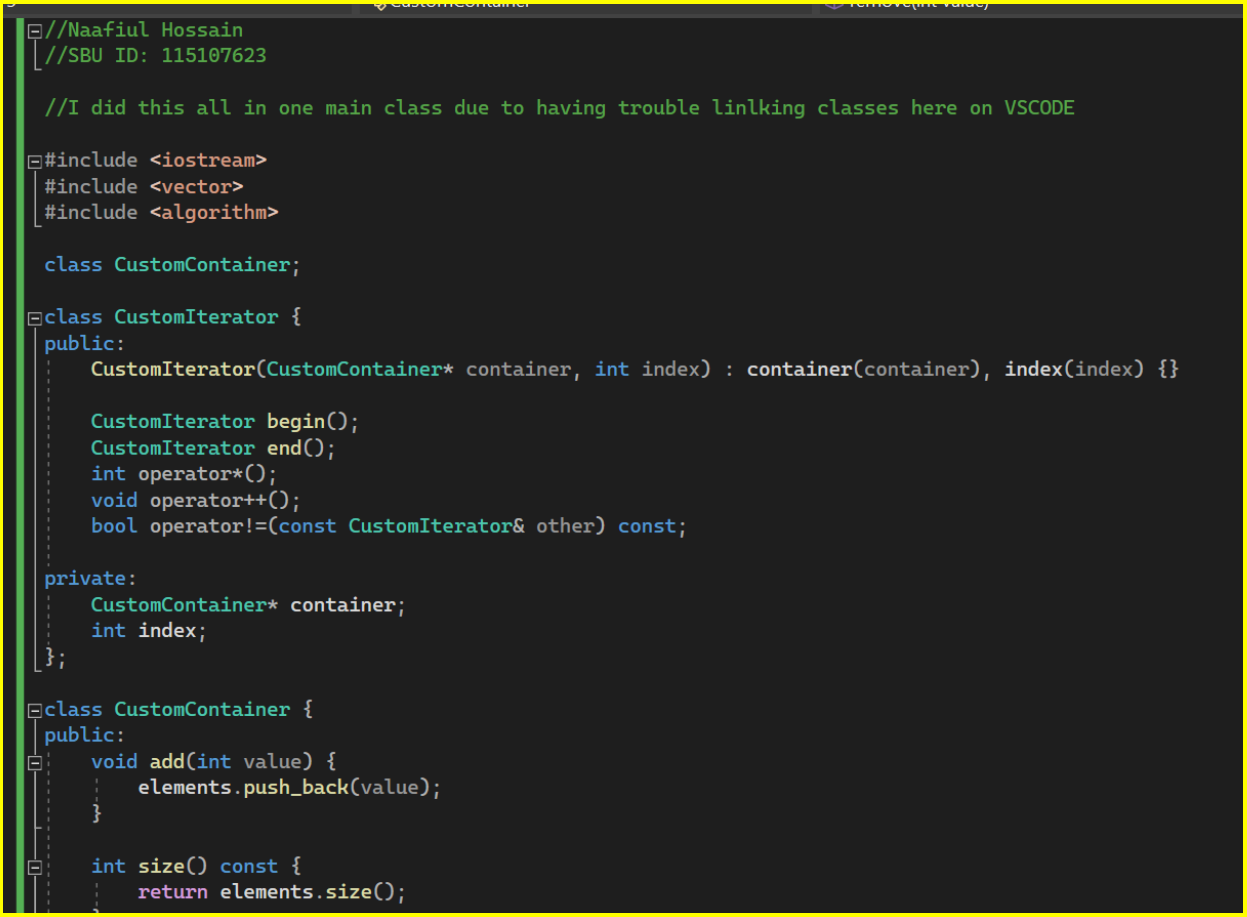
Screenshot of the Program running:



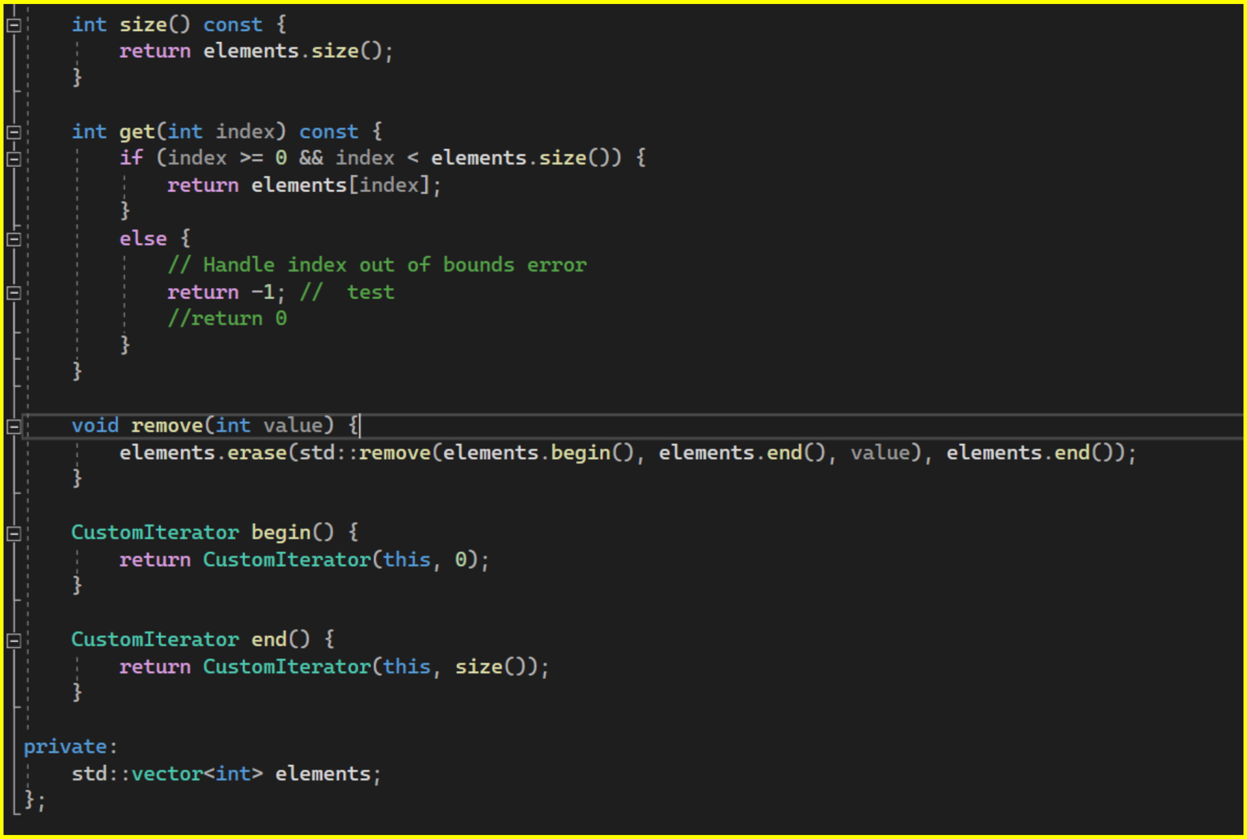
```
Testing the given test case:
Top: 2
Pop: 2
Is empty: false

C:\Users\Naafiul Hossain\Documents\Lab2\Lab1
 0.
To automatically close the console when debu
le when debugging stops.
Press any key to close this window . . .
```

Problem 5

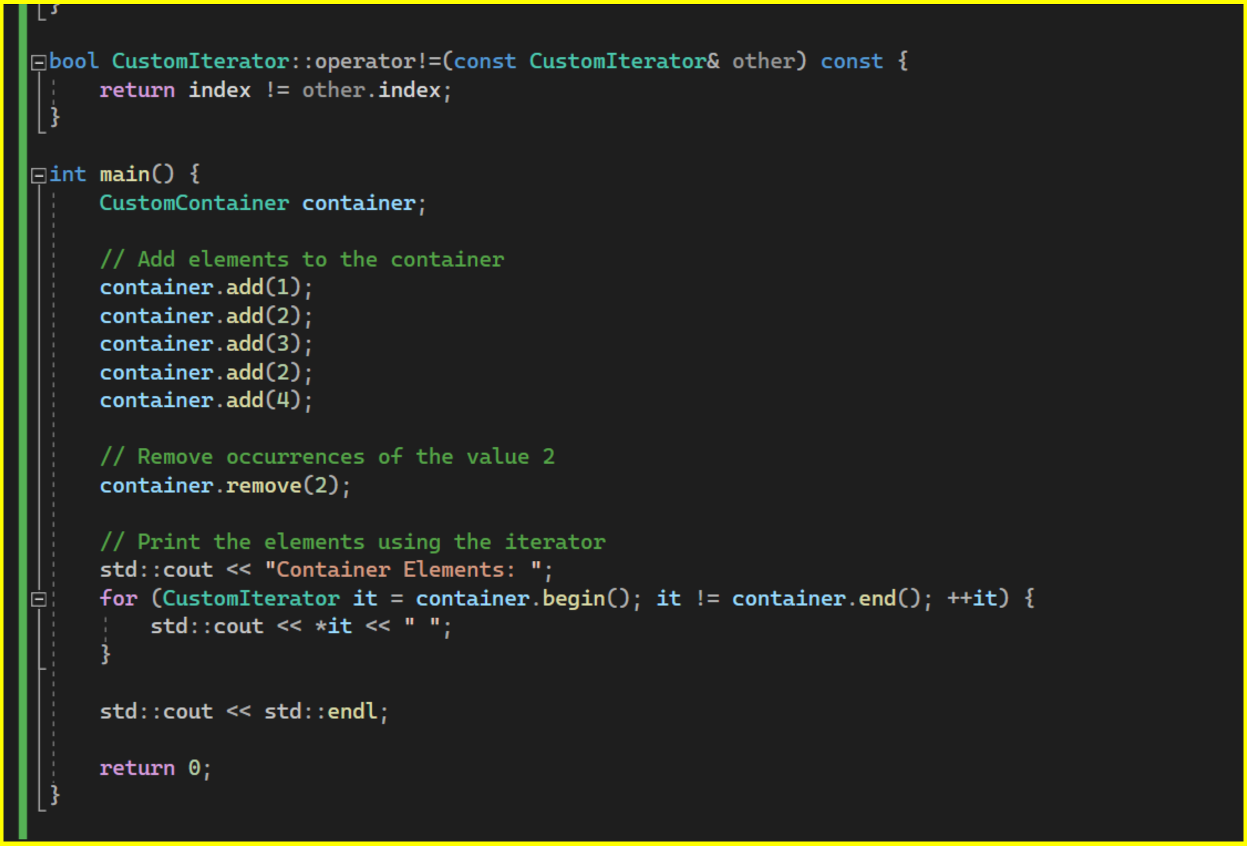
Main.cpp



```cpp
//Naafiul Hossain
//SBU ID: 115107623

//I did this all in one main class due to having trouble linlking classes here on VSCODE

#include <iostream>
#include <vector>
#include <algorithm>

class CustomContainer;

class CustomIterator {
public:
    CustomIterator(CustomContainer* container, int index) : container(container), index(index) {}

    CustomIterator begin();
    CustomIterator end();
    int operator*();
    void operator++();
    bool operator!=(const CustomIterator& other) const;

private:
    CustomContainer* container;
    int index;
};

class CustomContainer {
public:
    void add(int value) {
        elements.push_back(value);
    }

    int size() const {
        return elements.size();
```

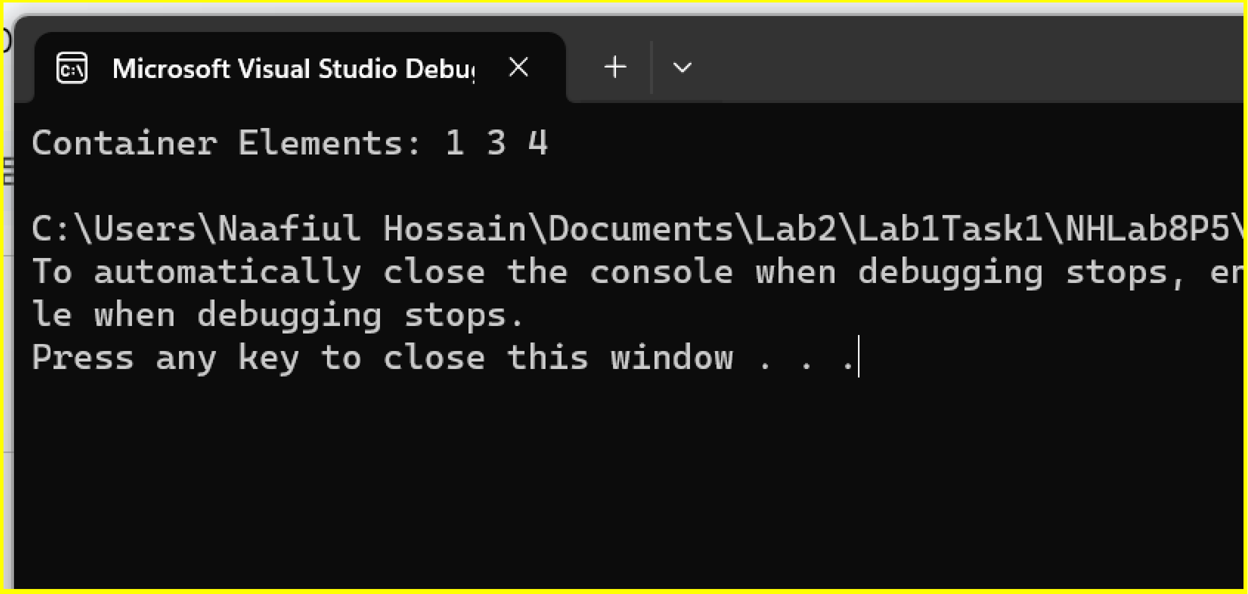


```cpp
    int size() const {
        return elements.size();
    }

    int get(int index) const {
        if (index >= 0 && index < elements.size()) {
            return elements[index];
        }
        else {
            // Handle index out of bounds error
            return -1; //  test
            //return 0
        }
    }

    void remove(int value) {
        elements.erase(std::remove(elements.begin(), elements.end(), value), elements.end());
    }

    CustomIterator begin() {
        return CustomIterator(this, 0);
    }

    CustomIterator end() {
        return CustomIterator(this, size());
    }

private:
    std::vector<int> elements;
};
```



```cpp
}

bool CustomIterator::operator!=(const CustomIterator& other) const {
    return index != other.index;
}

int main() {
    CustomContainer container;

    // Add elements to the container
    container.add(1);
    container.add(2);
    container.add(3);
    container.add(2);
    container.add(4);

    // Remove occurrences of the value 2
    container.remove(2);

    // Print the elements using the iterator
    std::cout << "Container Elements: ";
    for (CustomIterator it = container.begin(); it != container.end(); ++it) {
        std::cout << *it << " ";
    }

    std::cout << std::endl;

    return 0;
}
```

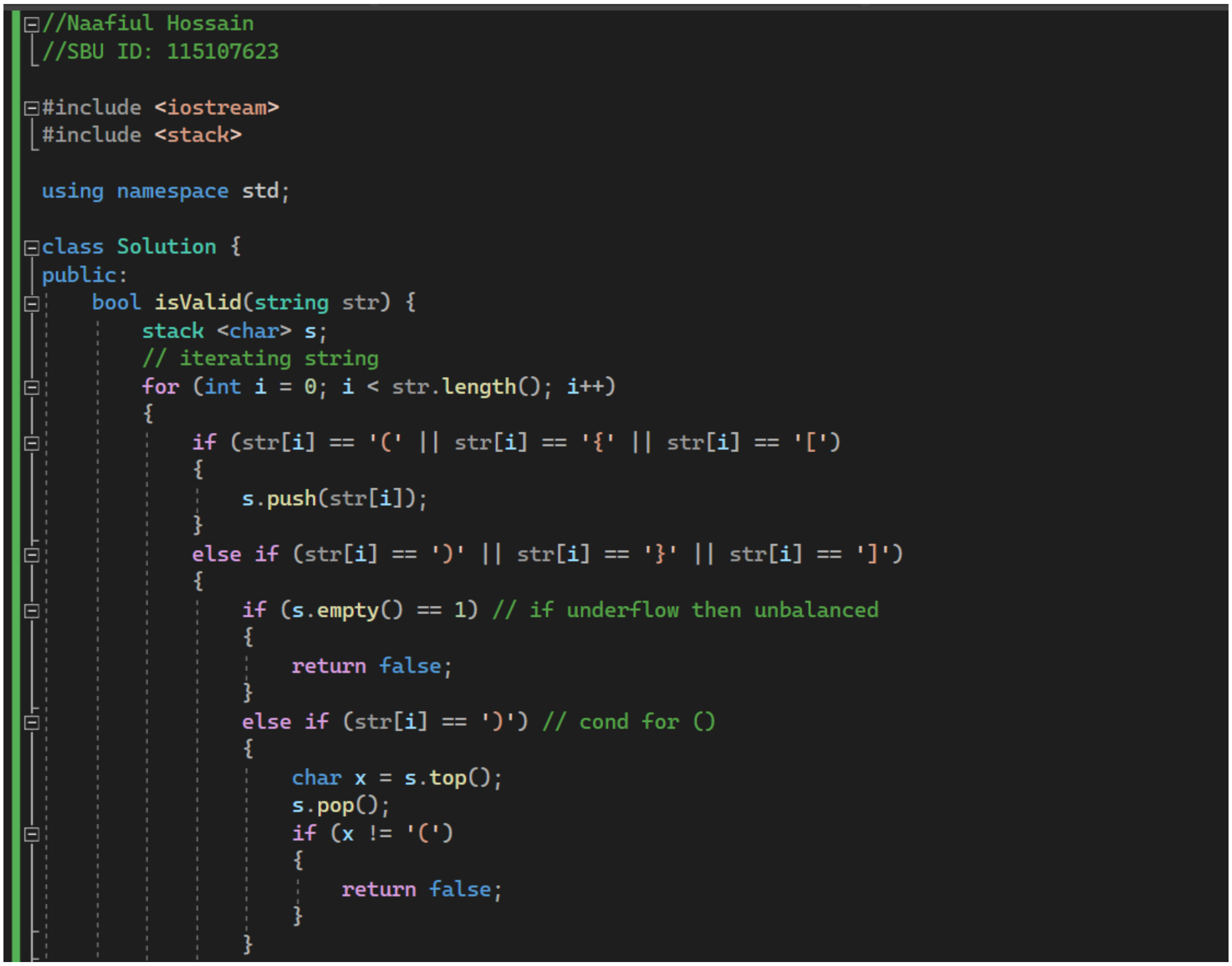
Screenshot of the running program:



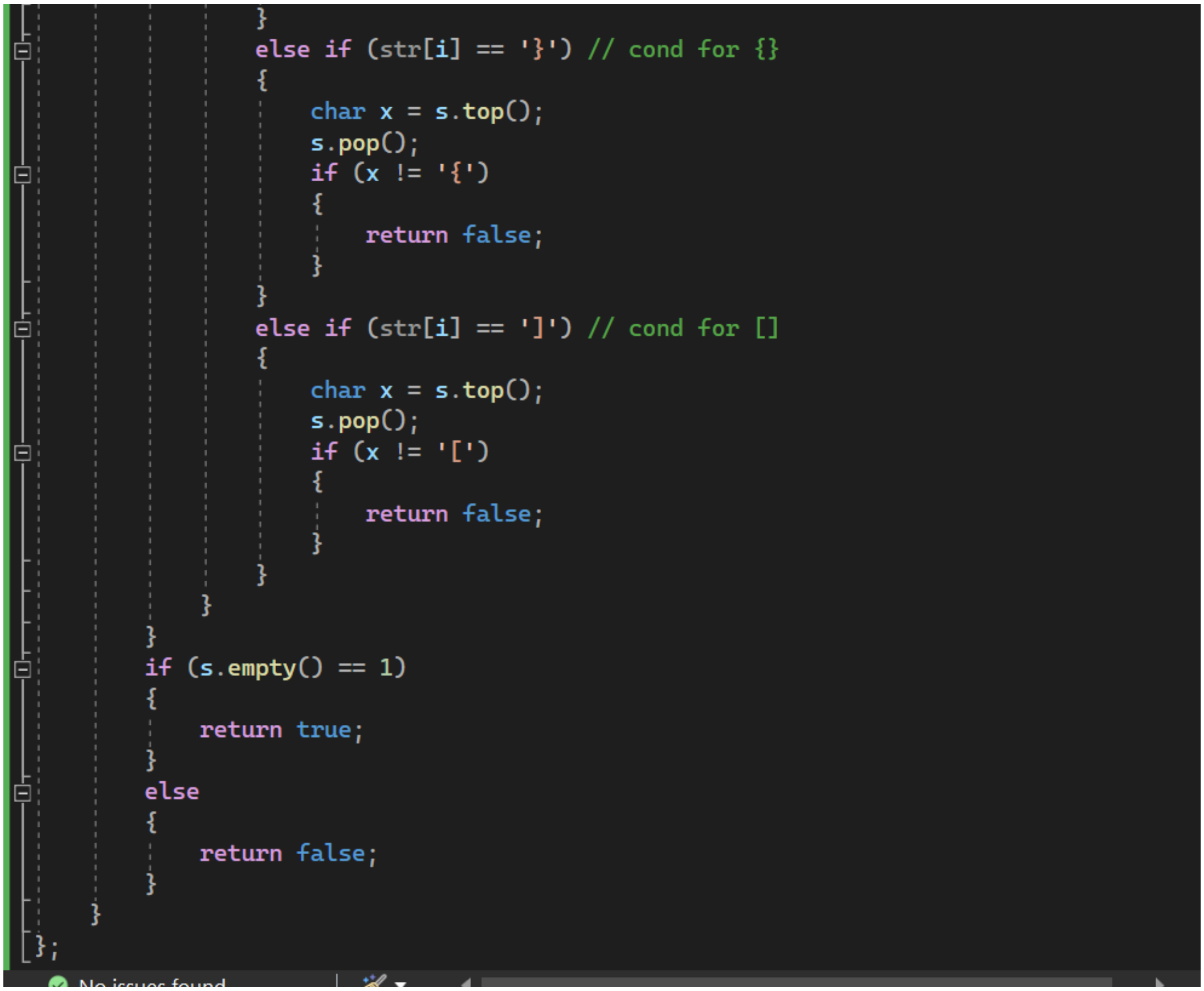
```
Container Elements: 1 3 4

C:\Users\Naafiul Hossain\Documents\Lab2\Lab1Task1\NHLab8P5\
To automatically close the console when debugging stops, er
le when debugging stops.
Press any key to close this window . . .
```

Problem 6 Extra Credit

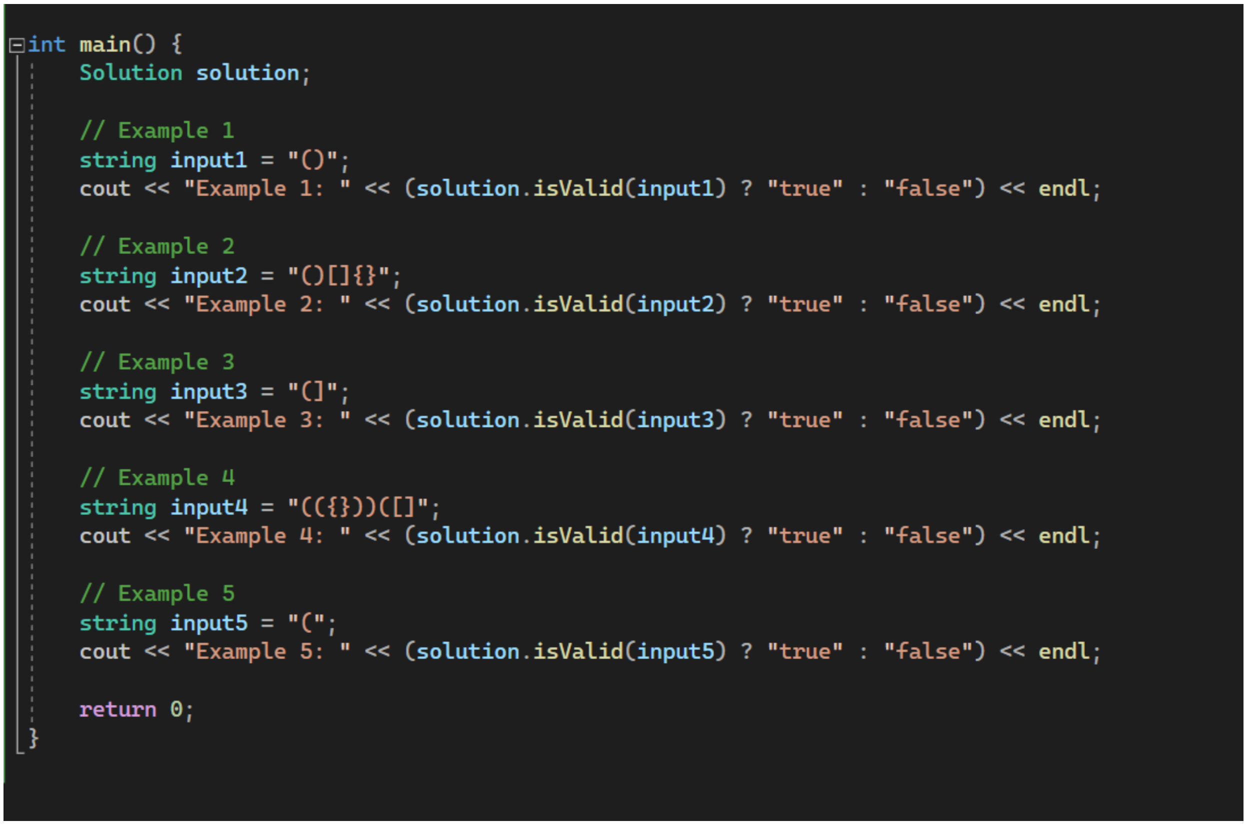
Main.cpp



```
//Naafiul Hossain
//SBU ID: 115107623

#include <iostream>
#include <stack>

using namespace std;

class Solution {
public:
    bool isValid(string str) {
        stack <char> s;
        // iterating string
        for (int i = 0; i < str.length(); i++)
        {
            if (str[i] == '(' || str[i] == '{' || str[i] == '[')
            {
                s.push(str[i]);
            }
            else if (str[i] == ')' || str[i] == '}' || str[i] == ']')
            {
                if (s.empty() == 1) // if underflow then unbalanced
                {
                    return false;
                }
                else if (str[i] == ')') // cond for ()
                {
                    char x = s.top();
                    s.pop();
                    if (x != '(')
                    {
                        return false;
                    }
                }
```

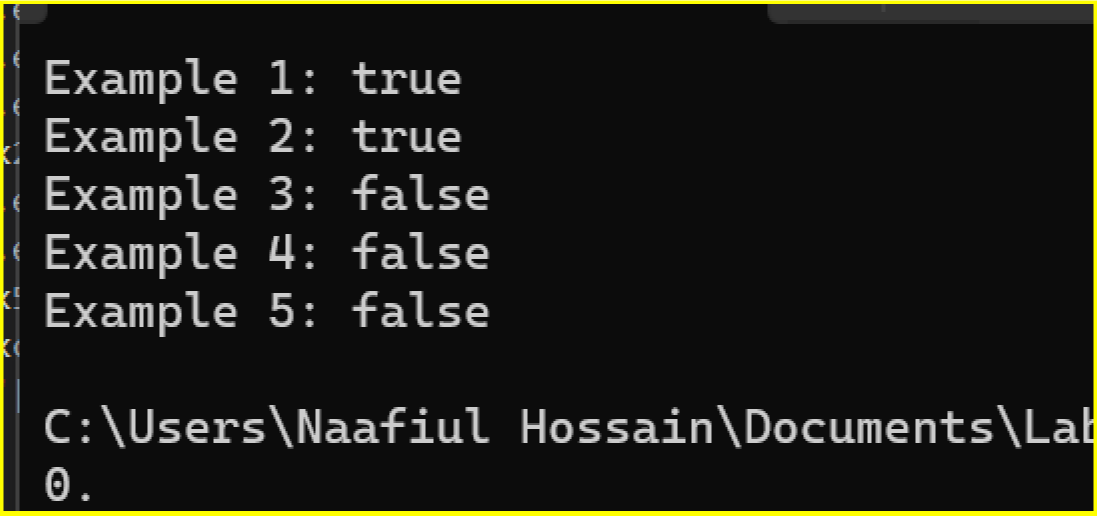


```
                }
                else if (str[i] == '}') // cond for {}
                {
                    char x = s.top();
                    s.pop();
                    if (x != '{')
                    {
                        return false;
                    }
                }
                else if (str[i] == ']') // cond for []
                {
                    char x = s.top();
                    s.pop();
                    if (x != '[')
                    {
                        return false;
                    }
                }
            }
        }
        if (s.empty() == 1)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
};
```



```
int main() {
    Solution solution;

    // Example 1
    string input1 = "()";
    cout << "Example 1: " << (solution.isValid(input1) ? "true" : "false") << endl;

    // Example 2
    string input2 = "()[]{}";
    cout << "Example 2: " << (solution.isValid(input2) ? "true" : "false") << endl;

    // Example 3
    string input3 = "(]";
    cout << "Example 3: " << (solution.isValid(input3) ? "true" : "false") << endl;

    // Example 4
    string input4 = "(({}))([]";
    cout << "Example 4: " << (solution.isValid(input4) ? "true" : "false") << endl;

    // Example 5
    string input5 = "(";
    cout << "Example 5: " << (solution.isValid(input5) ? "true" : "false") << endl;

    return 0;
}
```

Running solution:



```
Example 1: true
Example 2: true
Example 3: false
Example 4: false
Example 5: false

C:\Users\Naafiul Hossain\Documents\Lab
0.
```