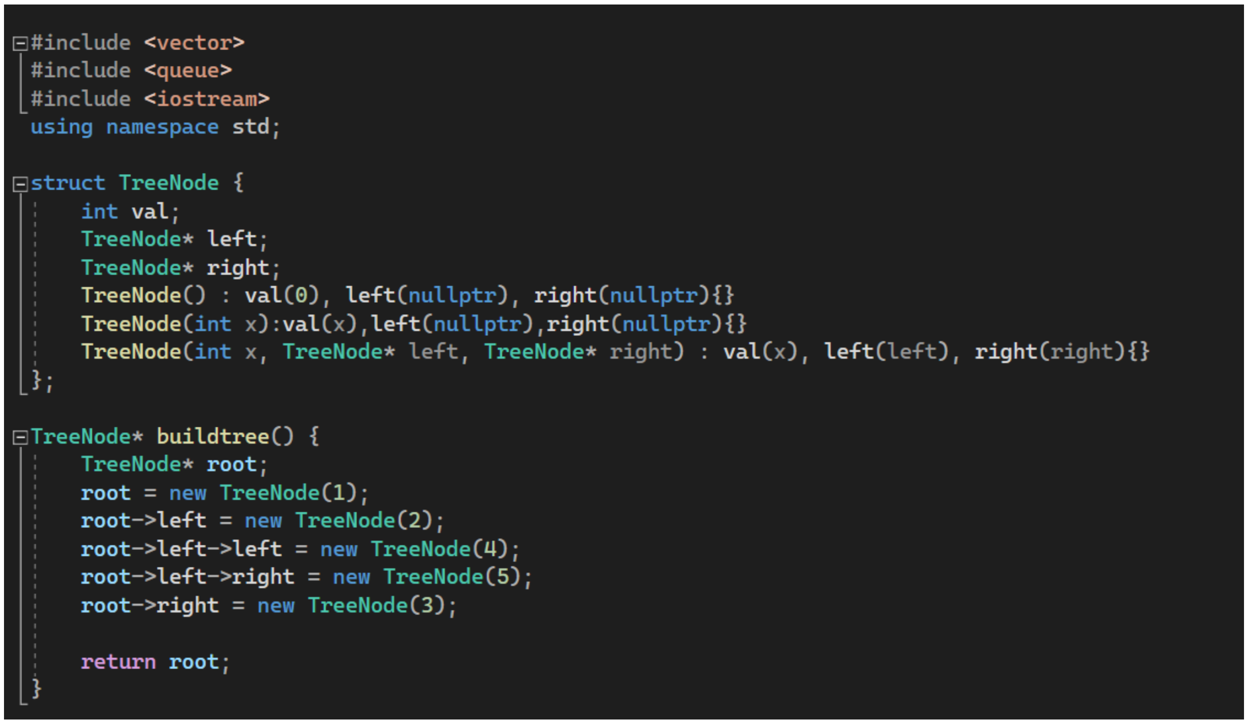
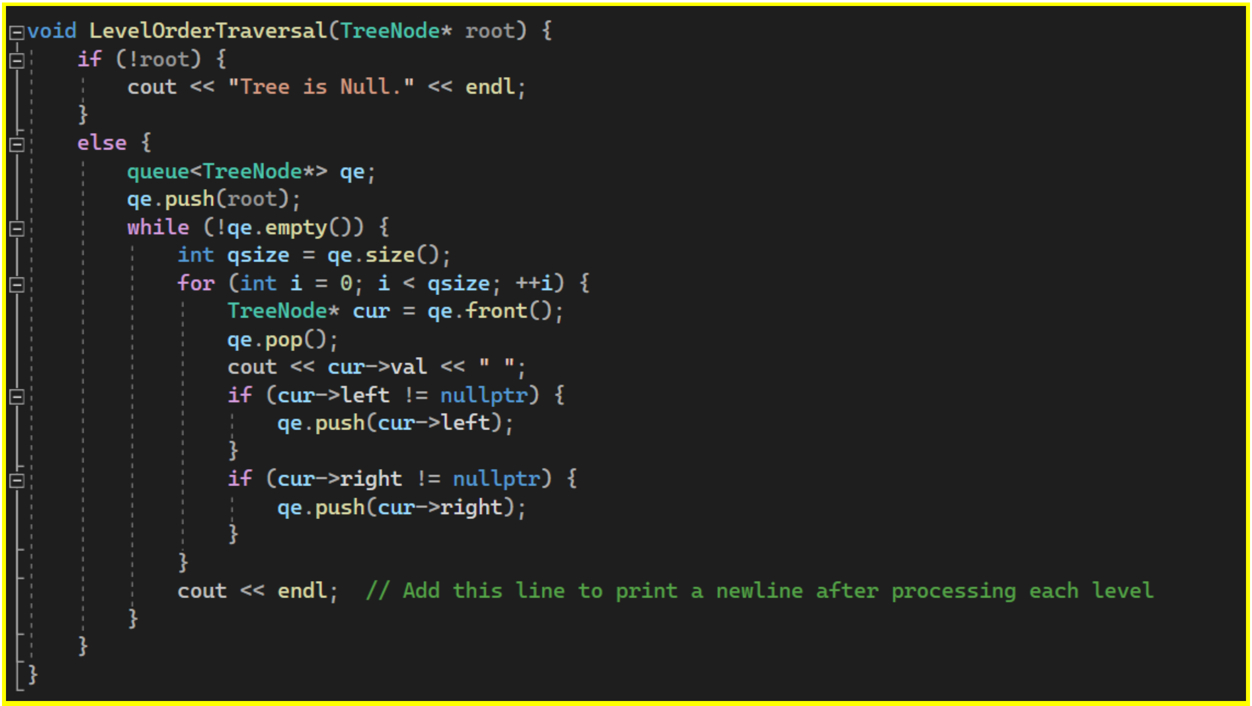
Naafiul Hossain
ESE224
115107623
Tuesday 10-12:50am

Problem 1:

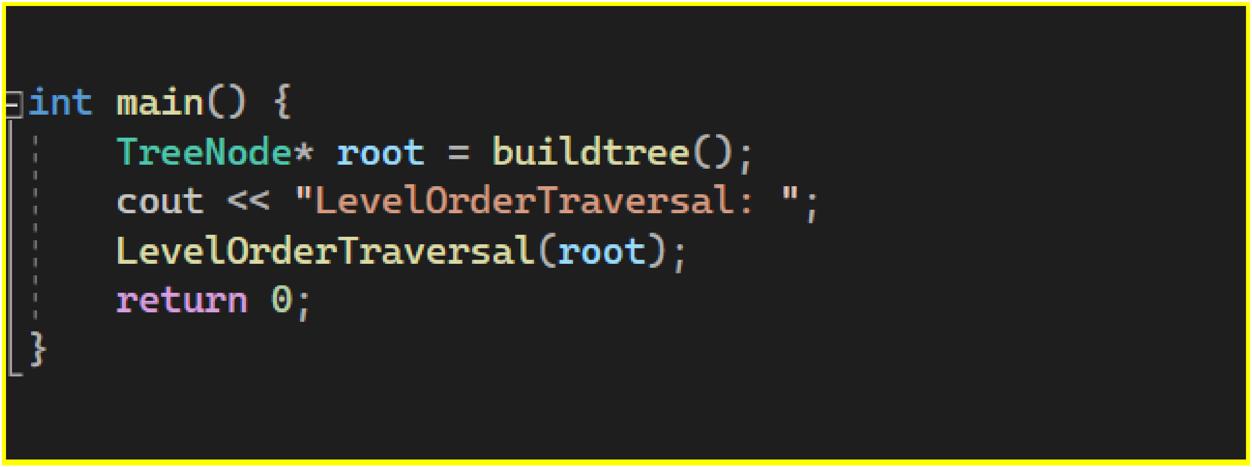
Main.cpp



```cpp
#include <vector>
#include <queue>
#include <iostream>
using namespace std;

struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;
    TreeNode() : val(0), left(nullptr), right(nullptr){}
    TreeNode(int x):val(x),left(nullptr),right(nullptr){}
    TreeNode(int x, TreeNode* left, TreeNode* right) : val(x), left(left), right(right){}
};

TreeNode* buildtree() {
    TreeNode* root;
    root = new TreeNode(1);
    root->left = new TreeNode(2);
    root->left->left = new TreeNode(4);
    root->left->right = new TreeNode(5);
    root->right = new TreeNode(3);

    return root;
}
```

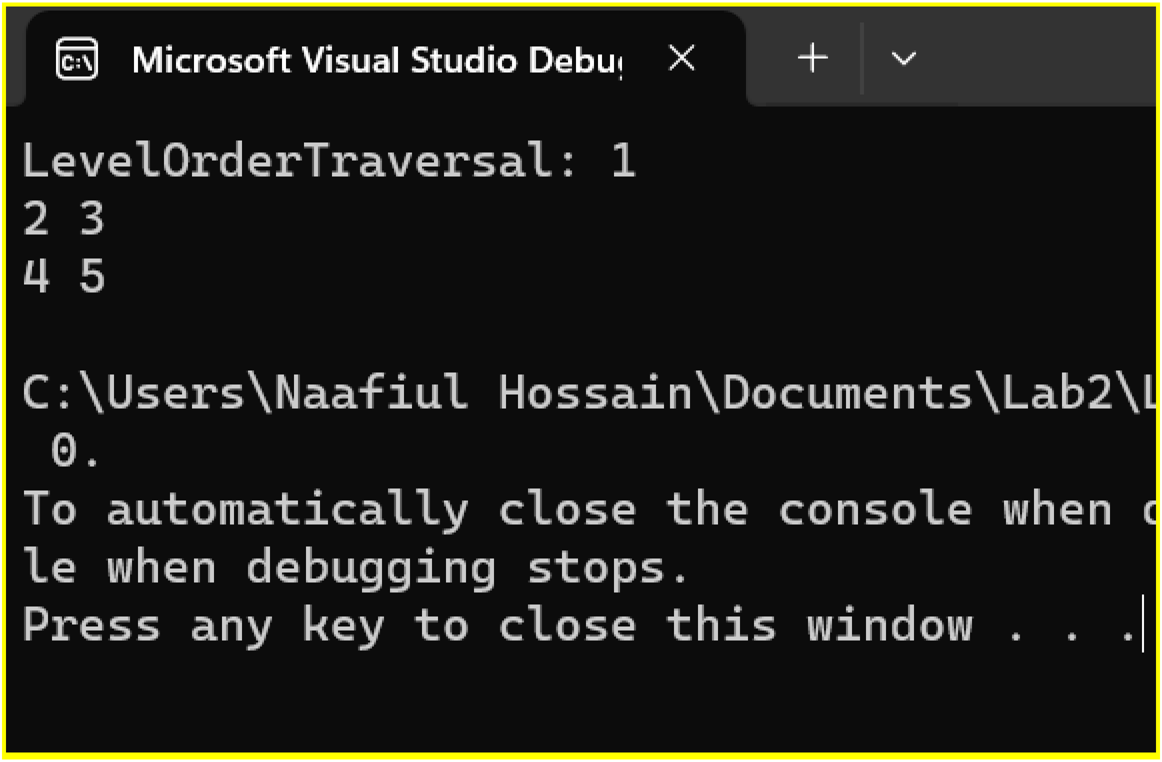


```cpp
void LevelOrderTraversal(TreeNode* root) {
    if (!root) {
        cout << "Tree is Null." << endl;
    }
    else {
        queue<TreeNode*> qe;
        qe.push(root);
        while (!qe.empty()) {
            int qsize = qe.size();
            for (int i = 0; i < qsize; ++i) {
                TreeNode* cur = qe.front();
                qe.pop();
                cout << cur->val << " ";
                if (cur->left != nullptr) {
                    qe.push(cur->left);
                }
                if (cur->right != nullptr) {
                    qe.push(cur->right);
                }
            }
            cout << endl;  // Add this line to print a newline after processing each level
        }
    }
}
```



```cpp
int main() {
    TreeNode* root = buildtree();
    cout << "LevelOrderTraversal: ";
    LevelOrderTraversal(root);
    return 0;
}
```

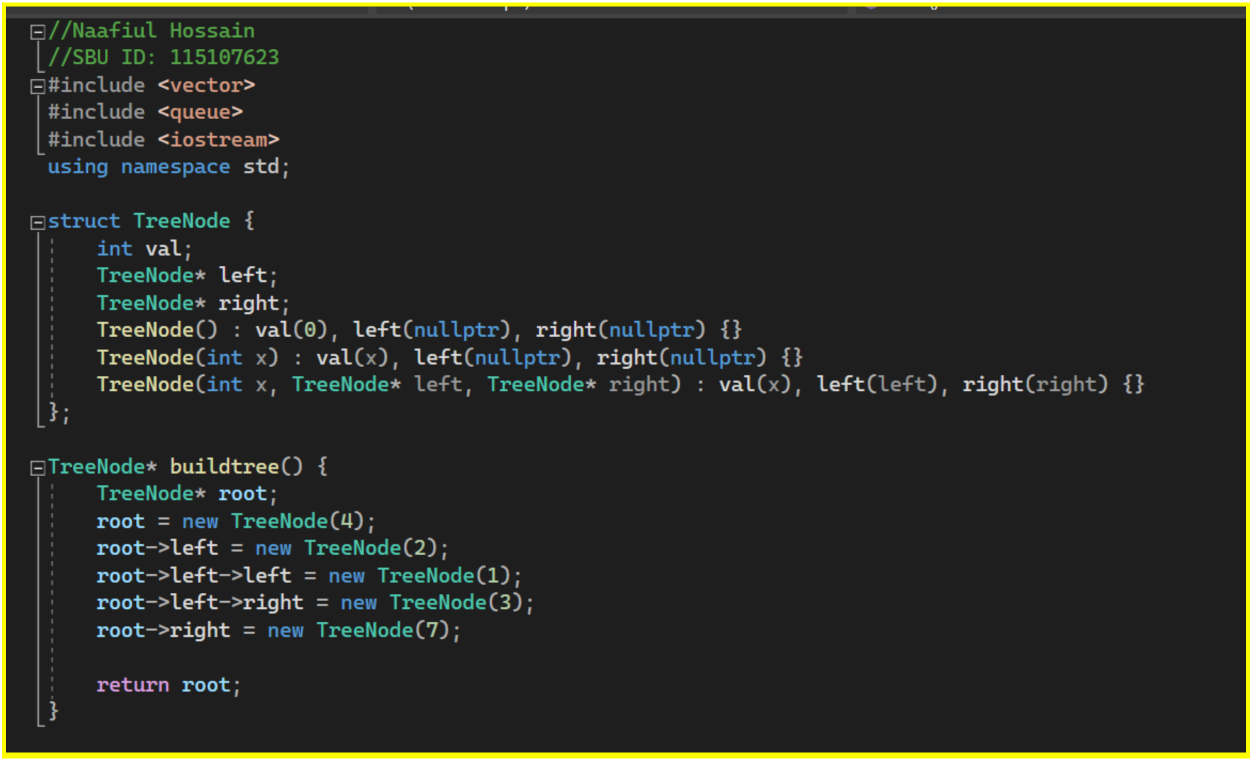
Screenshot of the running program:



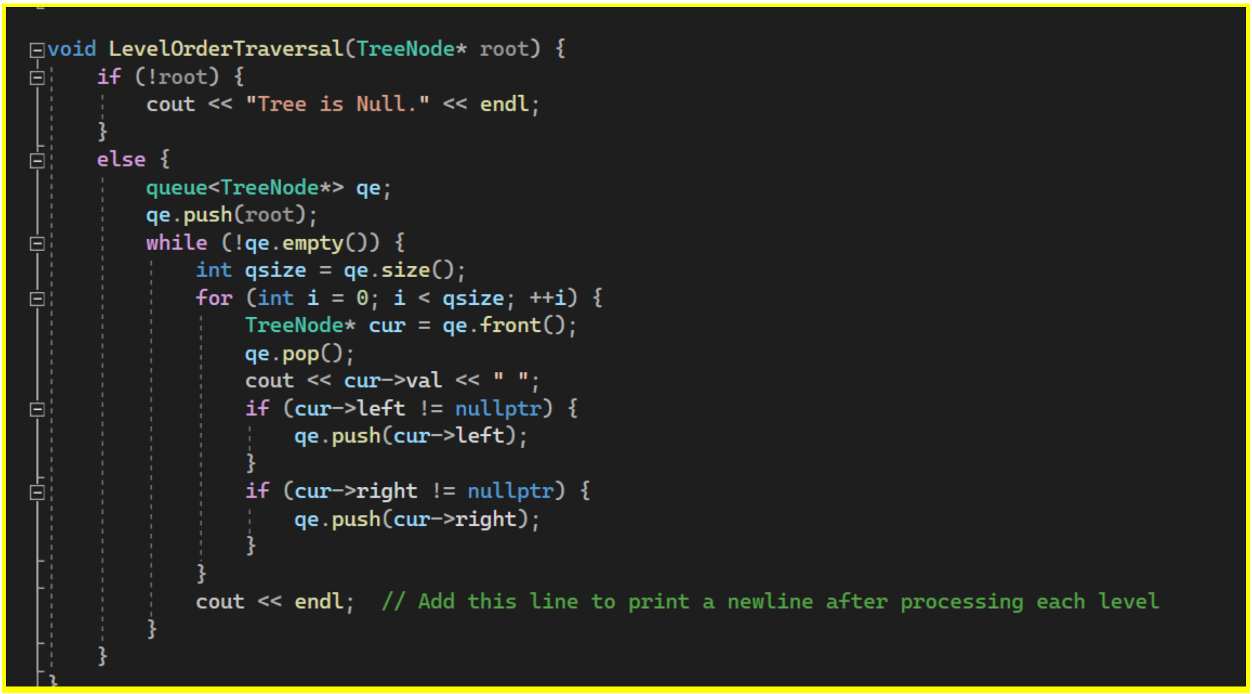
```
LevelOrderTraversal: 1
2 3
4 5

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

Problem 2

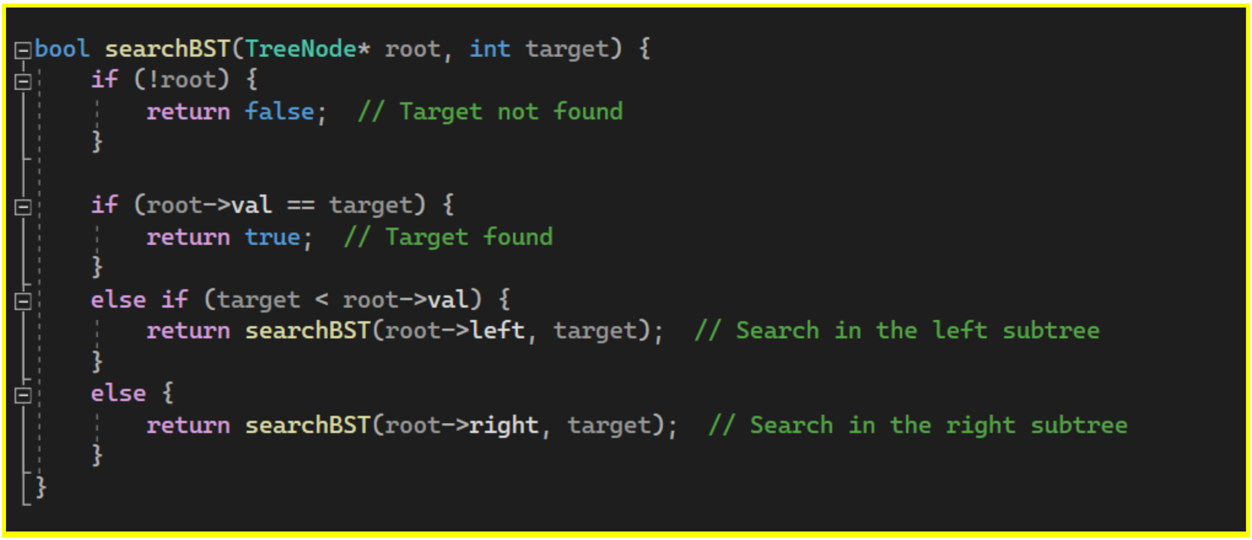
main.h



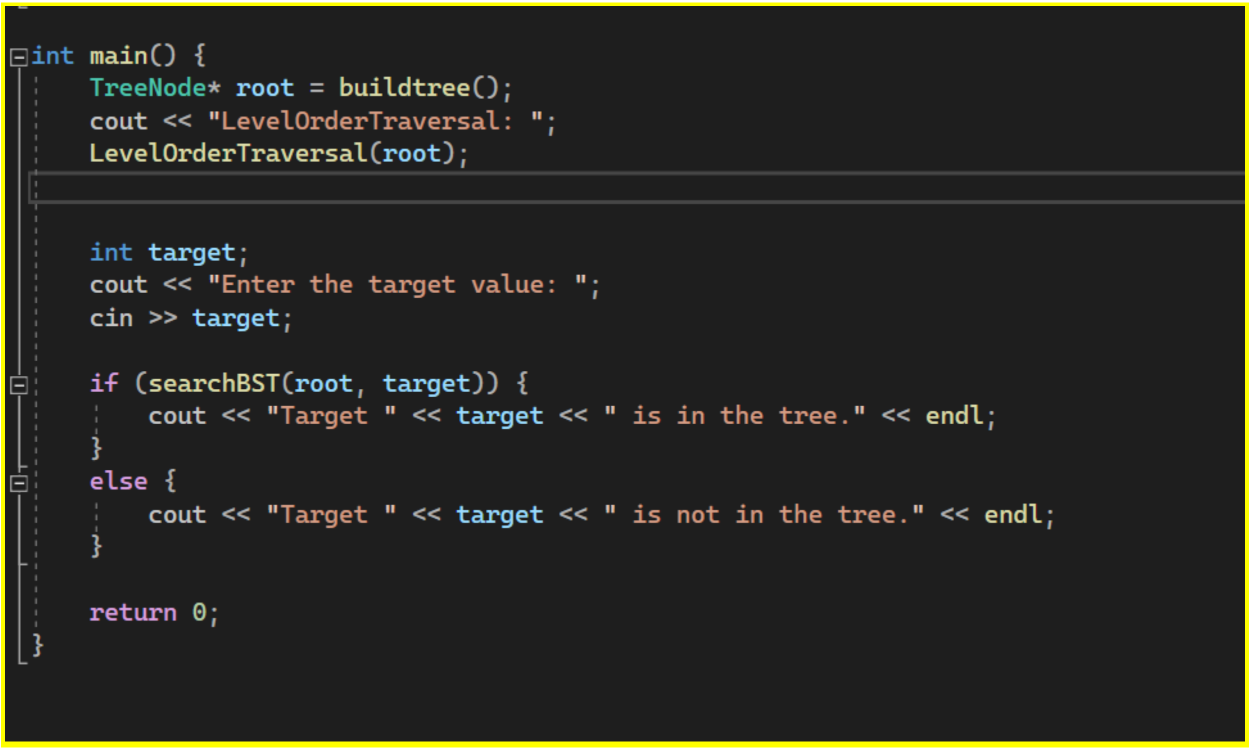
```cpp
//Naafiul Hossain
//SBU ID: 115107623
#include <vector>
#include <queue>
#include <iostream>
using namespace std;

struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;
    TreeNode() : val(0), left(nullptr), right(nullptr) {}
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
    TreeNode(int x, TreeNode* left, TreeNode* right) : val(x), left(left), right(right) {}
};

TreeNode* buildtree() {
    TreeNode* root;
    root = new TreeNode(4);
    root->left = new TreeNode(2);
    root->left->left = new TreeNode(1);
    root->left->right = new TreeNode(3);
    root->right = new TreeNode(7);

    return root;
}
```



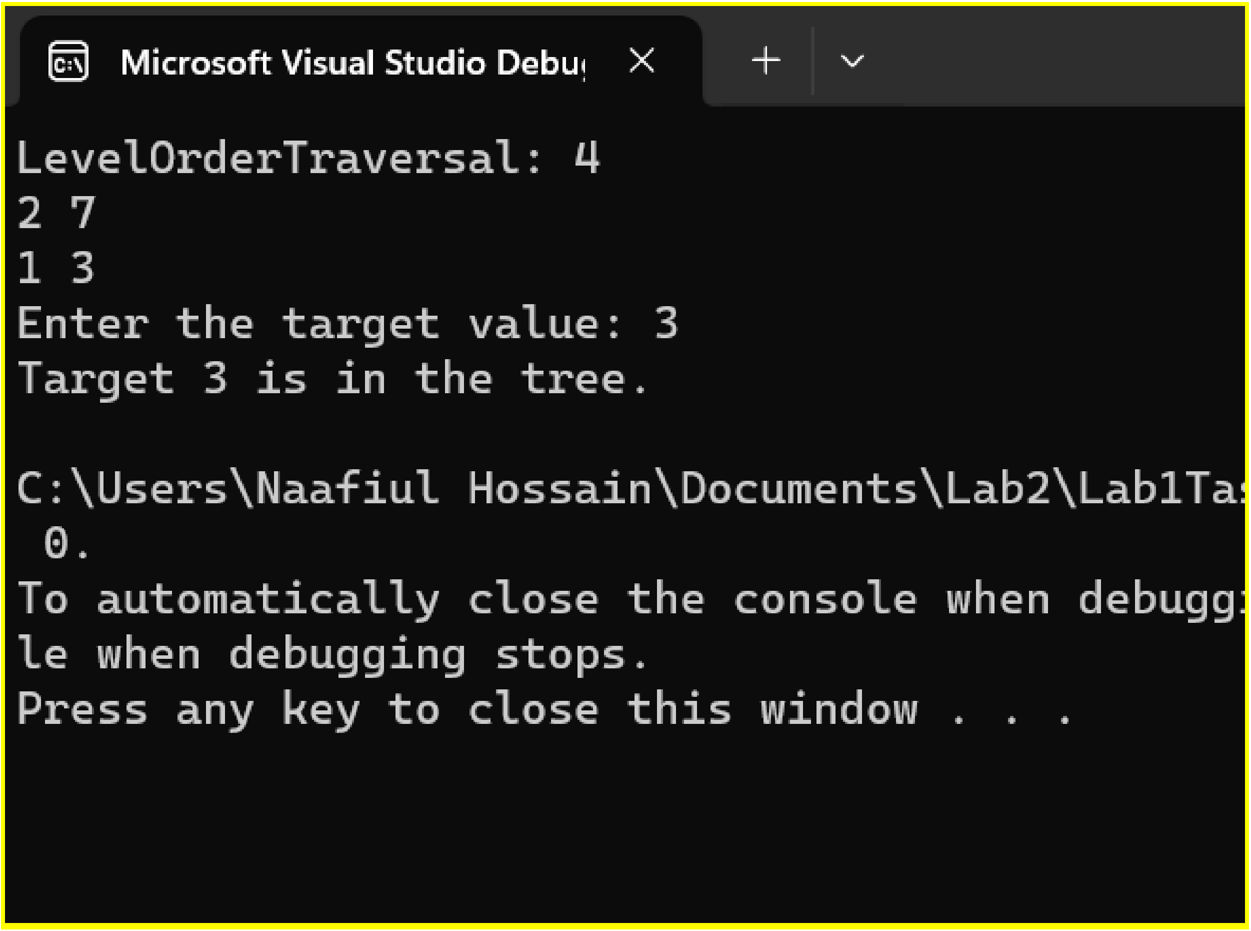
```cpp
void LevelOrderTraversal(TreeNode* root) {
    if (!root) {
        cout << "Tree is Null." << endl;
    }
    else {
        queue<TreeNode*> qe;
        qe.push(root);
        while (!qe.empty()) {
            int qsize = qe.size();
            for (int i = 0; i < qsize; ++i) {
                TreeNode* cur = qe.front();
                qe.pop();
                cout << cur->val << " ";
                if (cur->left != nullptr) {
                    qe.push(cur->left);
                }
                if (cur->right != nullptr) {
                    qe.push(cur->right);
                }
            }
            cout << endl;  // Add this line to print a newline after processing each level
        }
    }
}
```



```cpp
bool searchBST(TreeNode* root, int target) {
    if (!root) {
        return false;  // Target not found
    }

    if (root->val == target) {
        return true;  // Target found
    }
    else if (target < root->val) {
        return searchBST(root->left, target);  // Search in the left subtree
    }
    else {
        return searchBST(root->right, target);  // Search in the right subtree
    }
}
```


```cpp
int main() {
    TreeNode* root = buildtree();
    cout << "LevelOrderTraversal: ";
    LevelOrderTraversal(root);


    int target;
    cout << "Enter the target value: ";
    cin >> target;

    if (searchBST(root, target)) {
        cout << "Target " << target << " is in the tree." << endl;
    }
    else {
        cout << "Target " << target << " is not in the tree." << endl;
    }

    return 0;
}
```

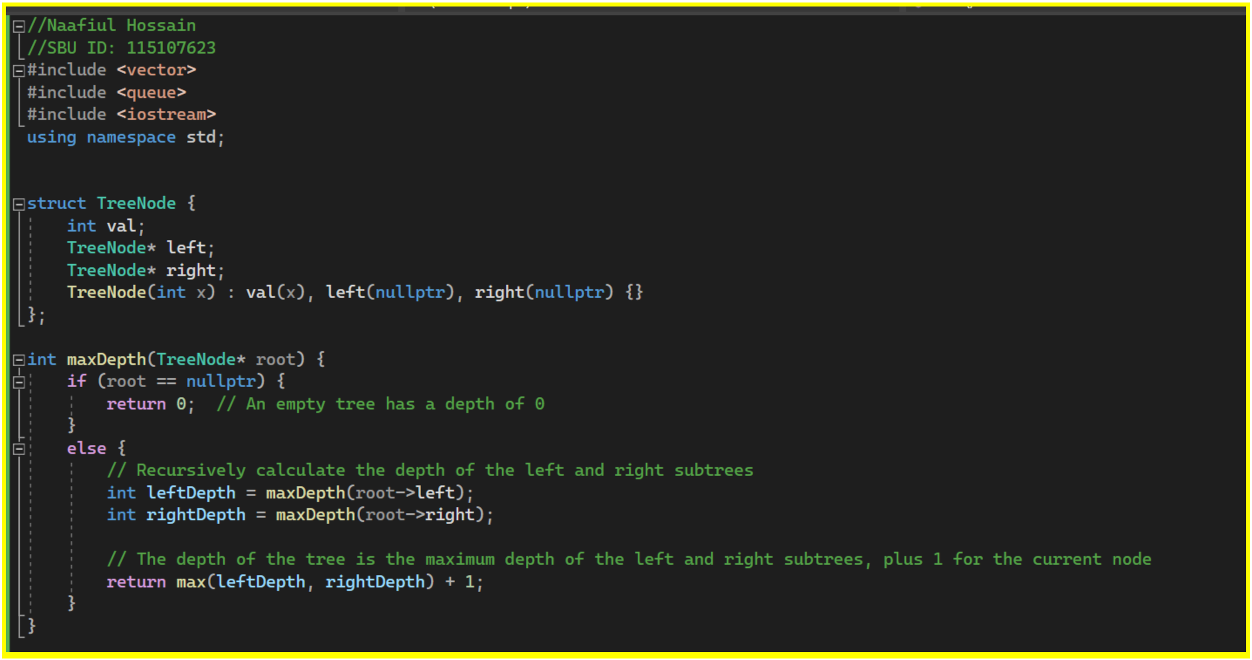
Running of the Program:



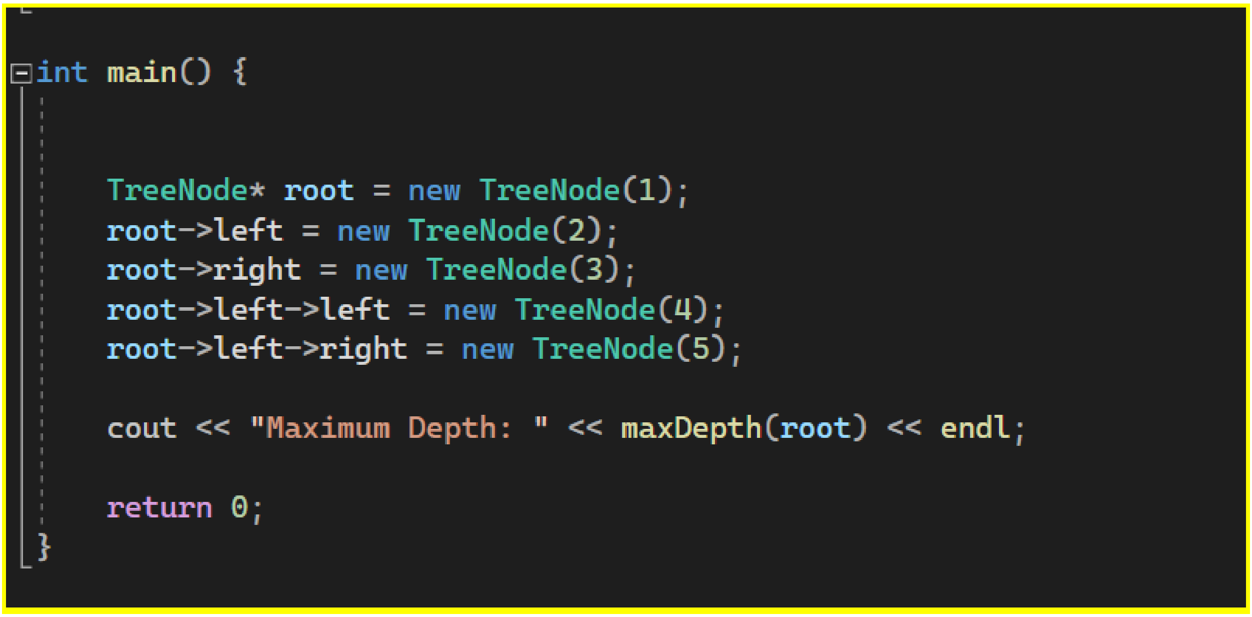
```
LevelOrderTraversal: 4
2 7
1 3
Enter the target value: 3
Target 3 is in the tree.

C:\Users\Naafiul Hossain\Documents\Lab2\Lab1Ta
 0.
To automatically close the console when debugg
le when debugging stops.
Press any key to close this window . . .
```

Problem 3

Main.cpp

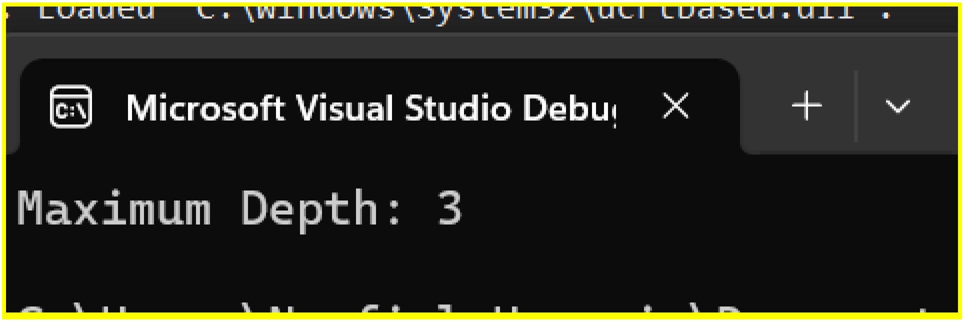


```
//Naafiul Hossain
//SBU ID: 115107623
#include <vector>
#include <queue>
#include <iostream>
using namespace std;


struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
};

int maxDepth(TreeNode* root) {
    if (root == nullptr) {
        return 0;  // An empty tree has a depth of 0
    }
    else {
        // Recursively calculate the depth of the left and right subtrees
        int leftDepth = maxDepth(root->left);
        int rightDepth = maxDepth(root->right);

        // The depth of the tree is the maximum depth of the left and right subtrees, plus 1 for the current node
        return max(leftDepth, rightDepth) + 1;
    }
}
```



```
int main() {


    TreeNode* root = new TreeNode(1);
    root->left = new TreeNode(2);
    root->right = new TreeNode(3);
    root->left->left = new TreeNode(4);
    root->left->right = new TreeNode(5);

    cout << "Maximum Depth: " << maxDepth(root) << endl;

    return 0;
}
```

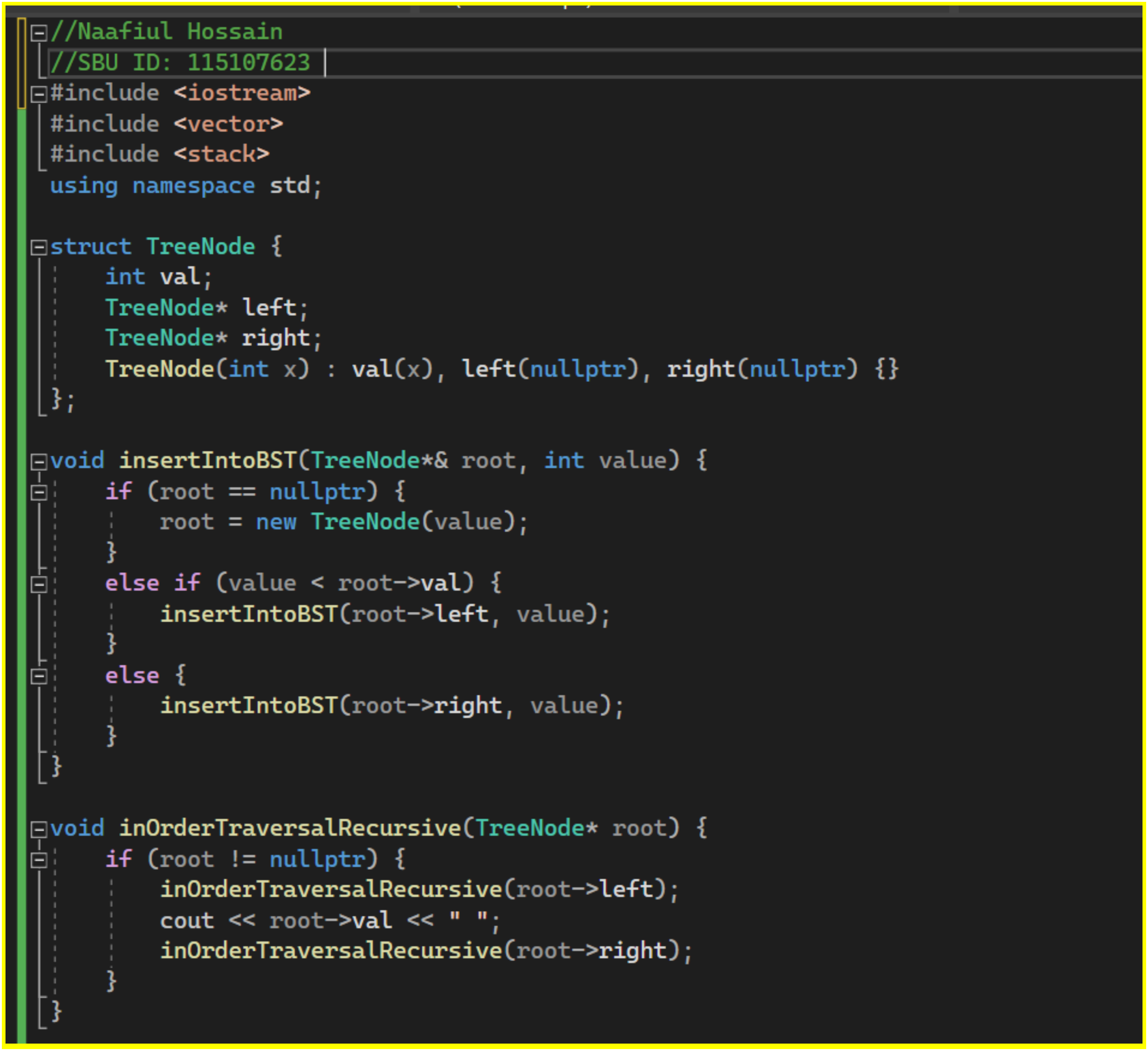
Screenshot of the running program:



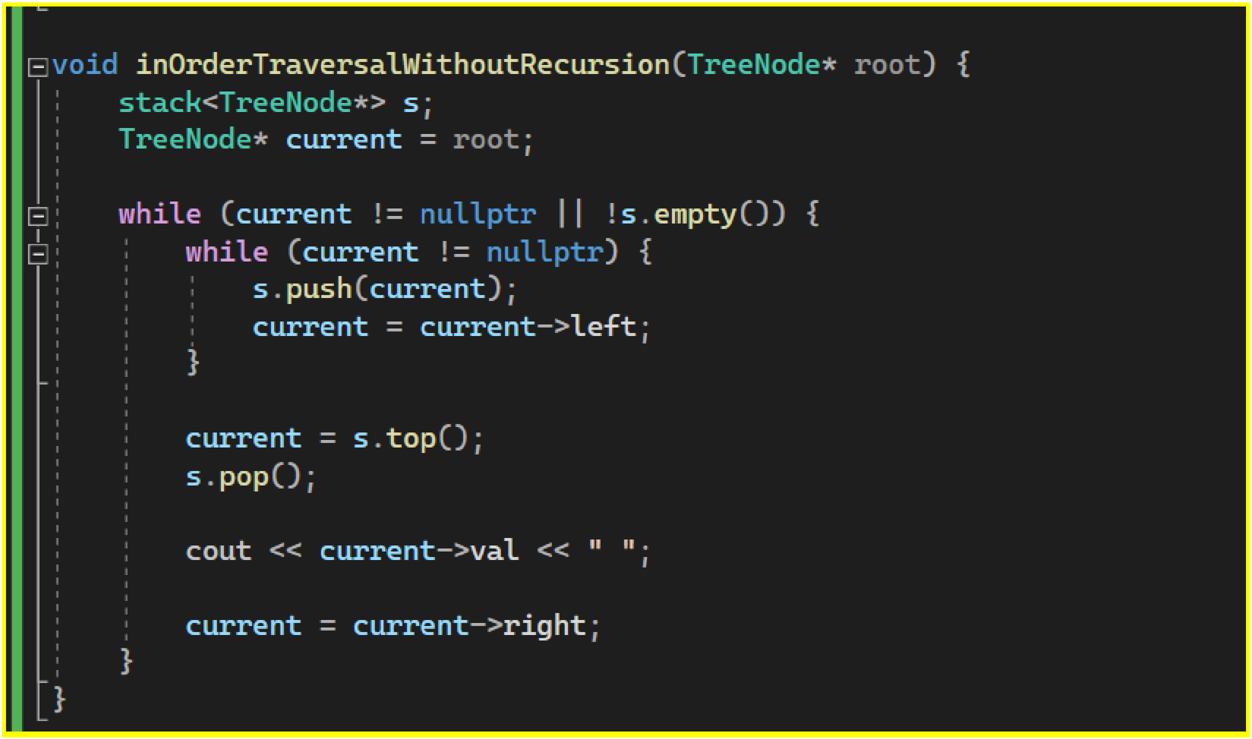
```
Maximum Depth: 3
```

Problem 4

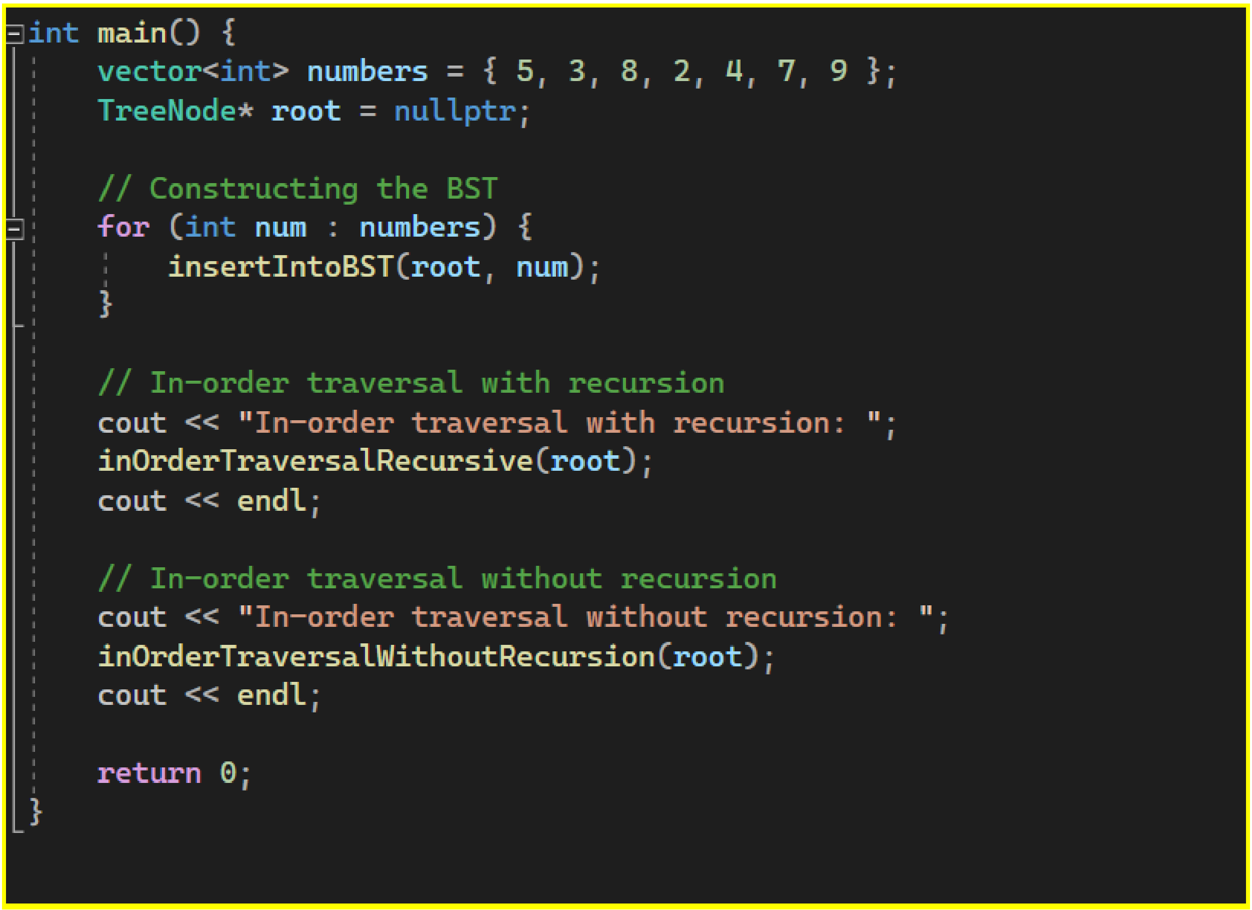
Main.cpp



```cpp
//Naafiul Hossain
//SBU ID: 115107623
#include <iostream>
#include <vector>
#include <stack>
using namespace std;

struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
};

void insertIntoBST(TreeNode*& root, int value) {
    if (root == nullptr) {
        root = new TreeNode(value);
    }
    else if (value < root->val) {
        insertIntoBST(root->left, value);
    }
    else {
        insertIntoBST(root->right, value);
    }
}

void inOrderTraversalRecursive(TreeNode* root) {
    if (root != nullptr) {
        inOrderTraversalRecursive(root->left);
        cout << root->val << " ";
        inOrderTraversalRecursive(root->right);
    }
}
```

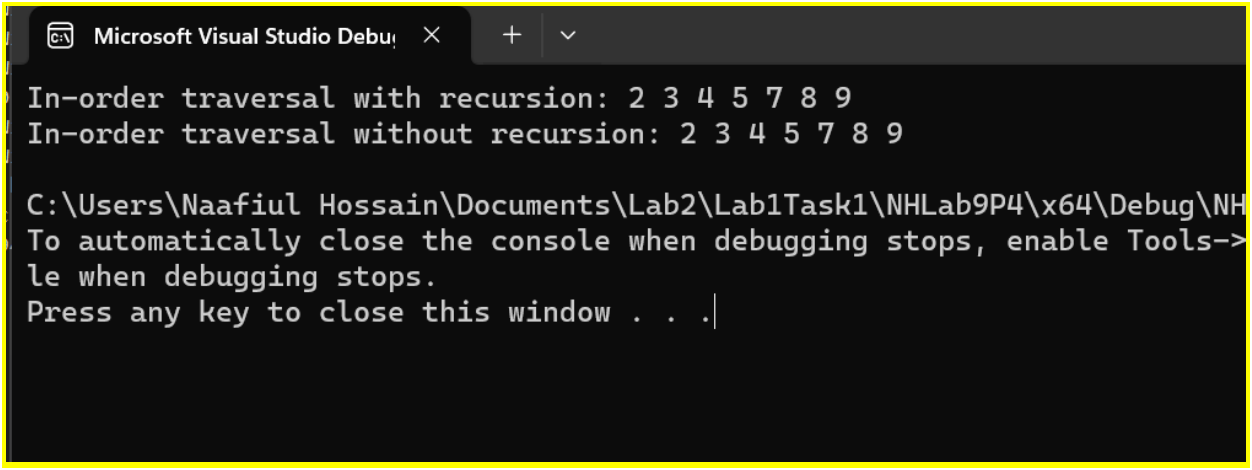


```cpp
void inOrderTraversalWithoutRecursion(TreeNode* root) {
    stack<TreeNode*> s;
    TreeNode* current = root;

    while (current != nullptr || !s.empty()) {
        while (current != nullptr) {
            s.push(current);
            current = current->left;
        }

        current = s.top();
        s.pop();

        cout << current->val << " ";

        current = current->right;
    }
}
```



```cpp
int main() {
    vector<int> numbers = { 5, 3, 8, 2, 4, 7, 9 };
    TreeNode* root = nullptr;

    // Constructing the BST
    for (int num : numbers) {
        insertIntoBST(root, num);
    }

    // In-order traversal with recursion
    cout << "In-order traversal with recursion: ";
    inOrderTraversalRecursive(root);
    cout << endl;

    // In-order traversal without recursion
    cout << "In-order traversal without recursion: ";
    inOrderTraversalWithoutRecursion(root);
    cout << endl;

    return 0;
}
```

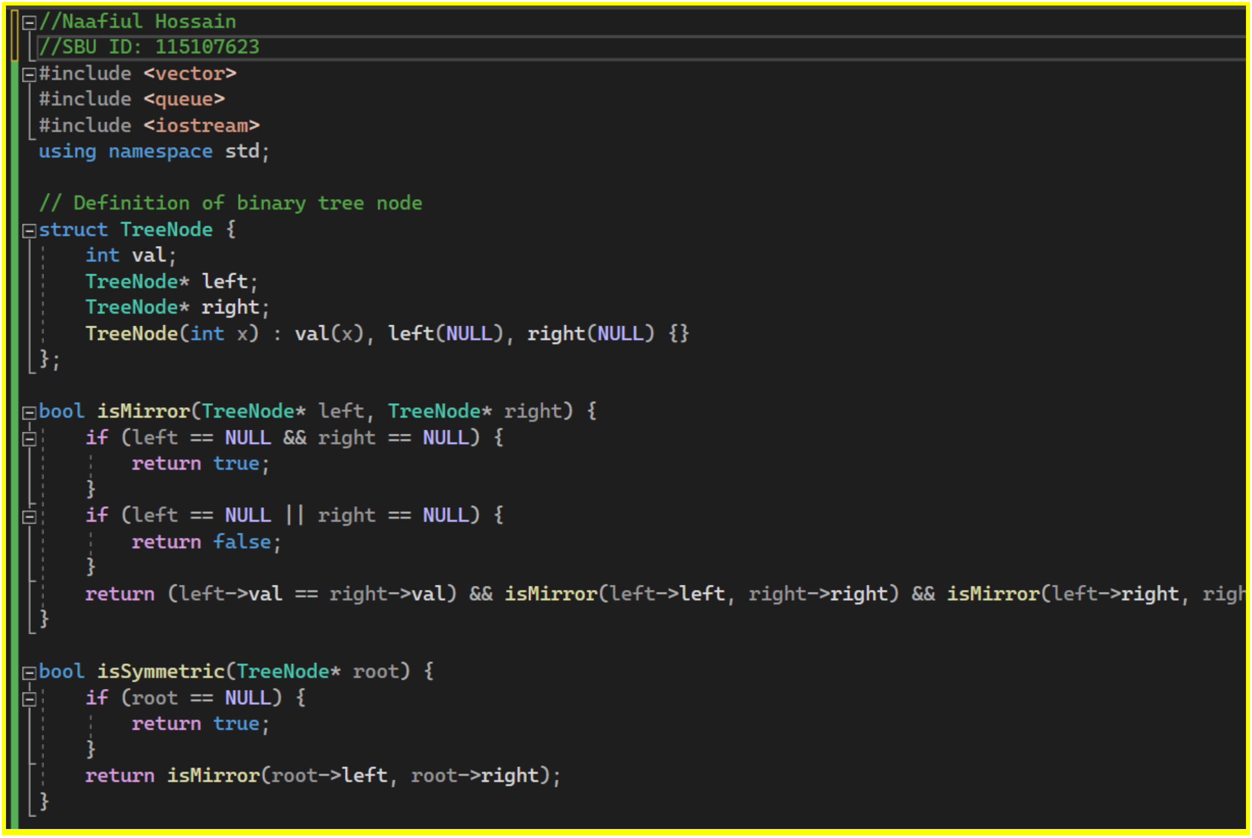
Screenshot of the Program running:


Microsoft Visual Studio Debu

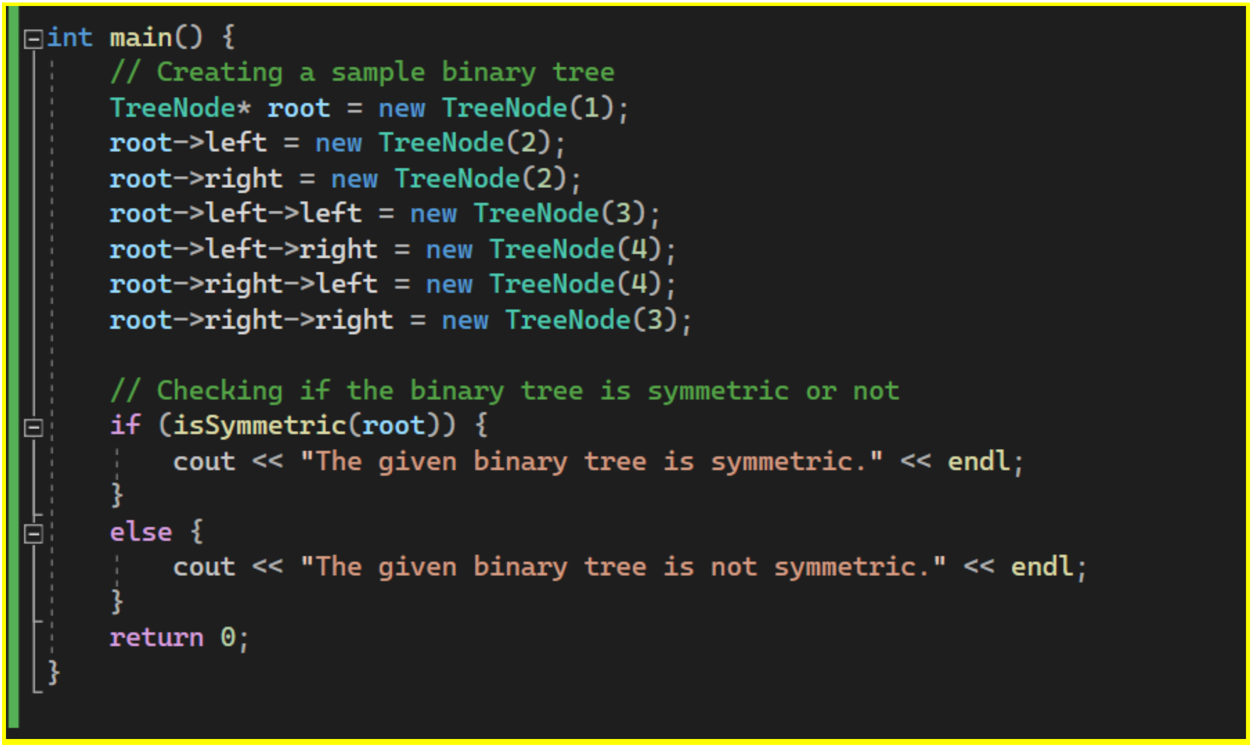
```
In-order traversal with recursion: 2 3 4 5 7 8 9
In-order traversal without recursion: 2 3 4 5 7 8 9

C:\Users\Naafiul Hossain\Documents\Lab2\Lab1Task1\NHLab9P4\x64\Debug\NH
To automatically close the console when debugging stops, enable Tools->
le when debugging stops.
Press any key to close this window . . .
```

Problem 5

Main.cpp

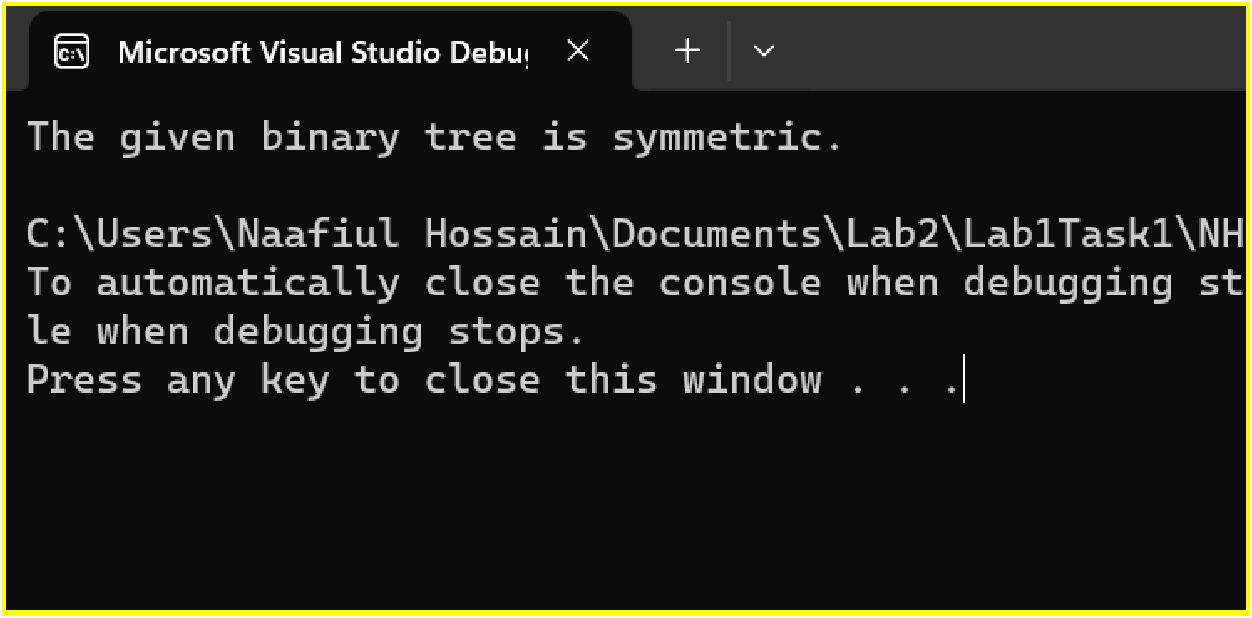


```cpp
//Naafiul Hossain
//SBU ID: 115107623
#include <vector>
#include <queue>
#include <iostream>
using namespace std;

// Definition of binary tree node
struct TreeNode {
    int val;
    TreeNode* left;
    TreeNode* right;
    TreeNode(int x) : val(x), left(NULL), right(NULL) {}
};

bool isMirror(TreeNode* left, TreeNode* right) {
    if (left == NULL && right == NULL) {
        return true;
    }
    if (left == NULL || right == NULL) {
        return false;
    }
    return (left->val == right->val) && isMirror(left->left, right->right) && isMirror(left->right, righ
}

bool isSymmetric(TreeNode* root) {
    if (root == NULL) {
        return true;
    }
    return isMirror(root->left, root->right);
}
```



```cpp
int main() {
    // Creating a sample binary tree
    TreeNode* root = new TreeNode(1);
    root->left = new TreeNode(2);
    root->right = new TreeNode(2);
    root->left->left = new TreeNode(3);
    root->left->right = new TreeNode(4);
    root->right->left = new TreeNode(4);
    root->right->right = new TreeNode(3);

    // Checking if the binary tree is symmetric or not
    if (isSymmetric(root)) {
        cout << "The given binary tree is symmetric." << endl;
    }
    else {
        cout << "The given binary tree is not symmetric." << endl;
    }
    return 0;
}
```

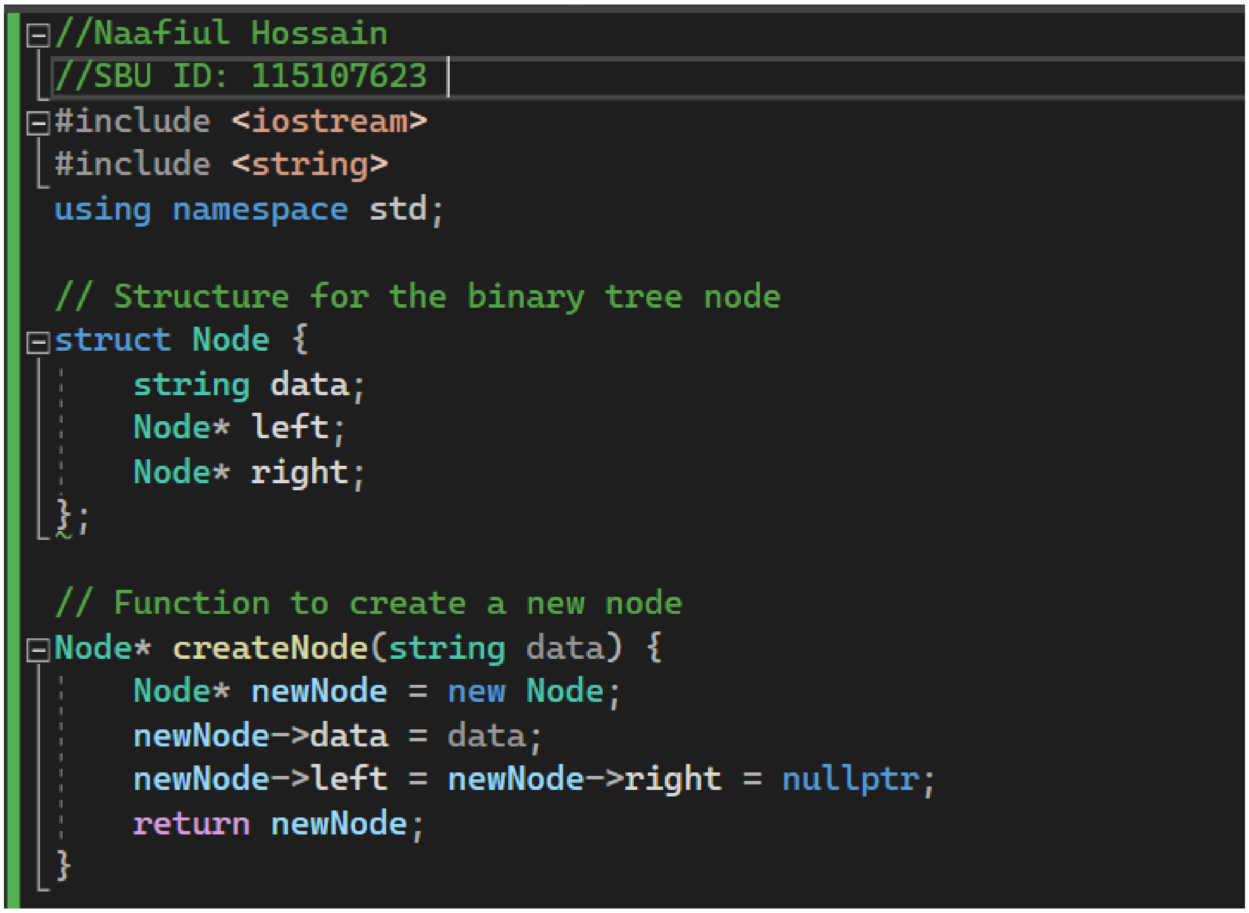
Screenshot of the running program:



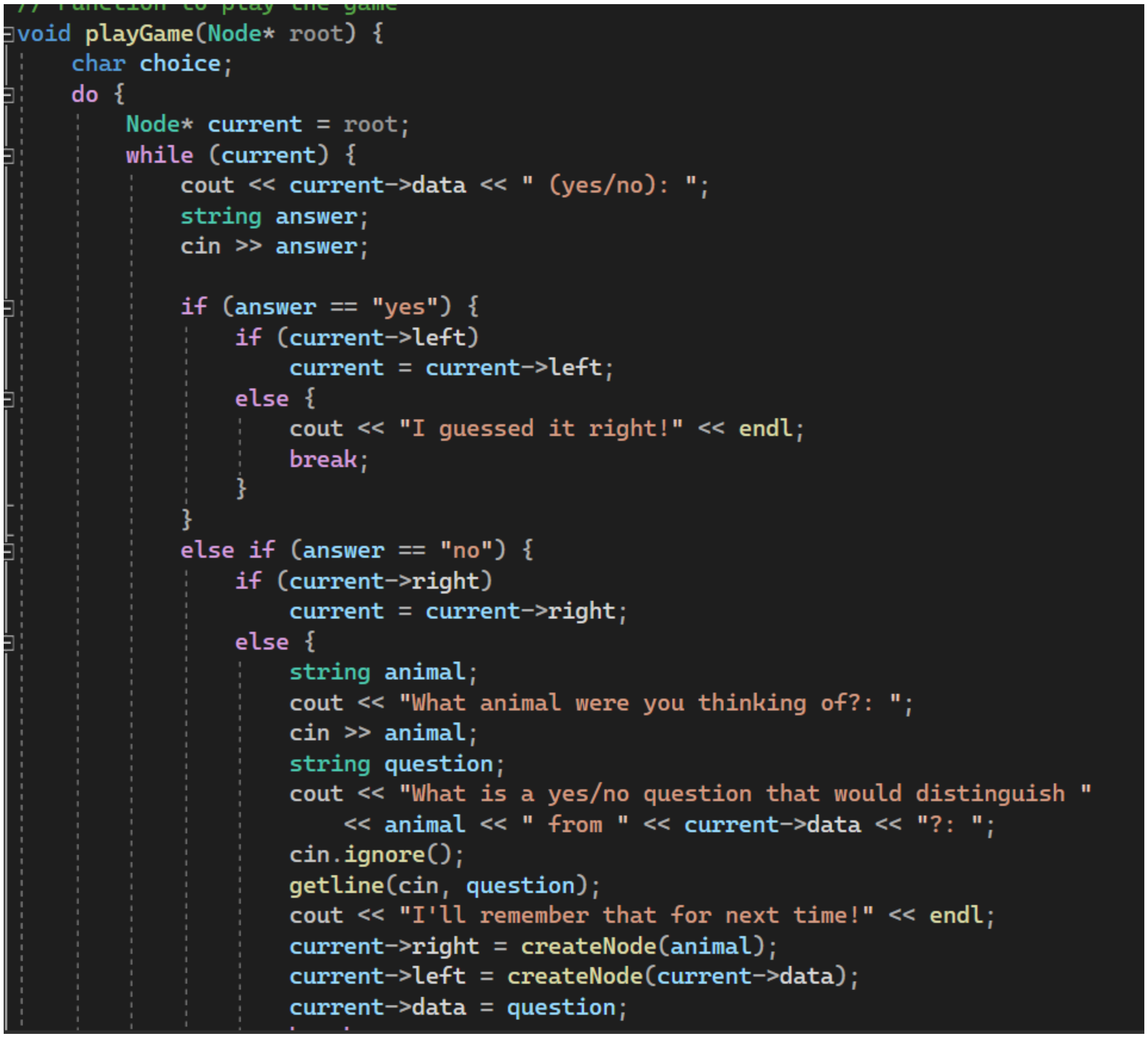
```
The given binary tree is symmetric.

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

Problem 6 Extra Credit

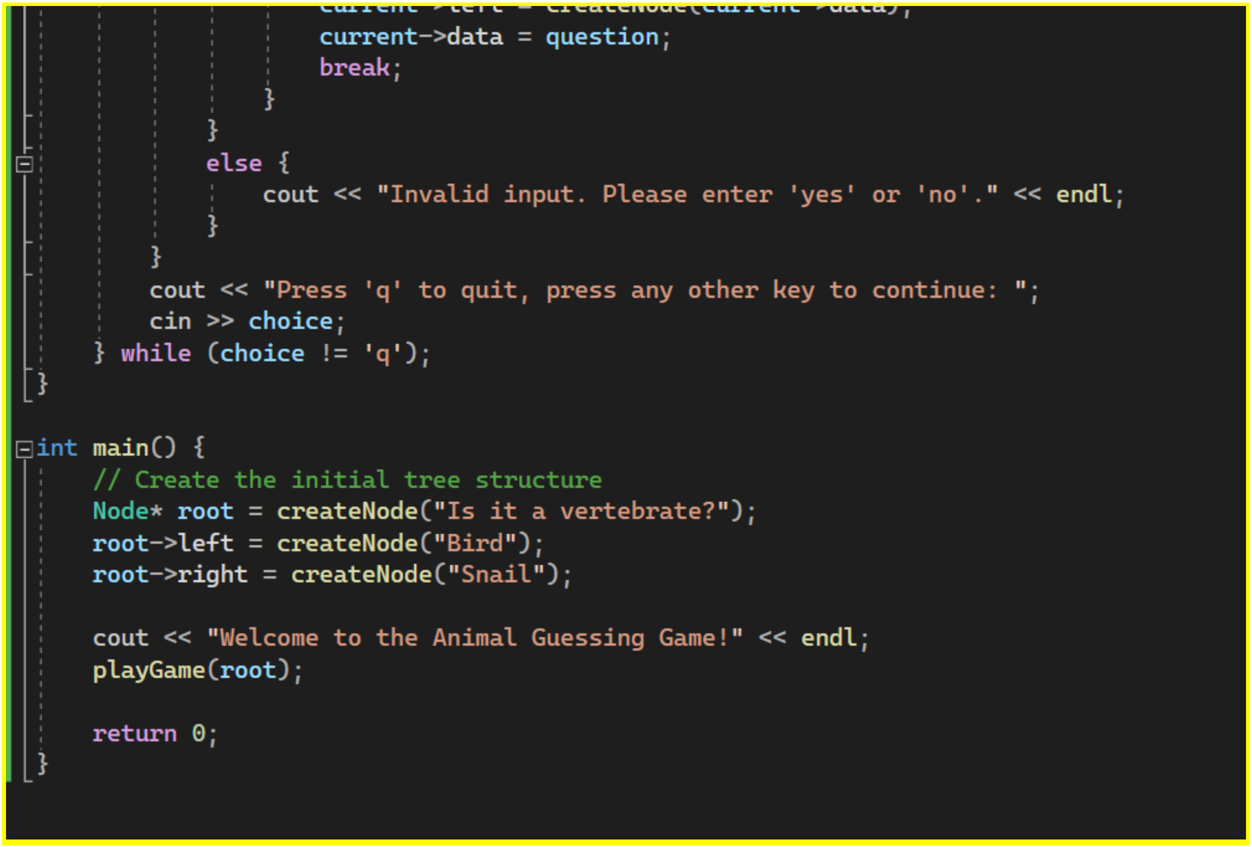
Main.cpp



```cpp
//Naafiul Hossain
//SBU ID: 115107623
#include <iostream>
#include <string>
using namespace std;

// Structure for the binary tree node
struct Node {
    string data;
    Node* left;
    Node* right;
};

// Function to create a new node
Node* createNode(string data) {
    Node* newNode = new Node;
    newNode->data = data;
    newNode->left = newNode->right = nullptr;
    return newNode;
}
```

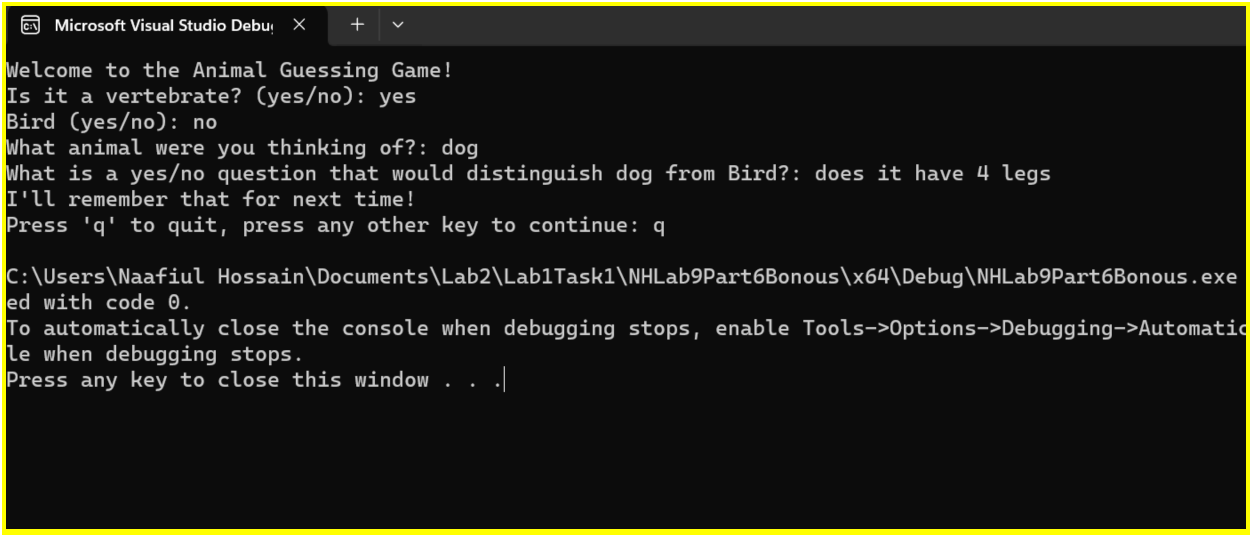


```cpp
// Function to play the game
void playGame(Node* root) {
    char choice;
    do {
        Node* current = root;
        while (current) {
            cout << current->data << " (yes/no): ";
            string answer;
            cin >> answer;

            if (answer == "yes") {
                if (current->left)
                    current = current->left;
                else {
                    cout << "I guessed it right!" << endl;
                    break;
                }
            }
            else if (answer == "no") {
                if (current->right)
                    current = current->right;
                else {
                    string animal;
                    cout << "What animal were you thinking of?: ";
                    cin >> animal;
                    string question;
                    cout << "What is a yes/no question that would distinguish "
                        << animal << " from " << current->data << "?: ";
                    cin.ignore();
                    getline(cin, question);
                    cout << "I'll remember that for next time!" << endl;
                    current->right = createNode(animal);
                    current->left = createNode(current->data);
                    current->data = question;
```



```
                    current->data = question;
                    break;
                }
            }
            else {
                cout << "Invalid input. Please enter 'yes' or 'no'." << endl;
            }
        }
        cout << "Press 'q' to quit, press any other key to continue: ";
        cin >> choice;
    } while (choice != 'q');
}

int main() {
    // Create the initial tree structure
    Node* root = createNode("Is it a vertebrate?");
    root->left = createNode("Bird");
    root->right = createNode("Snail");

    cout << "Welcome to the Animal Guessing Game!" << endl;
    playGame(root);

    return 0;
}
```

Running solution:



```
Welcome to the Animal Guessing Game!
Is it a vertebrate? (yes/no): yes
Bird (yes/no): no
What animal were you thinking of?: dog
What is a yes/no question that would distinguish dog from Bird?: does it have 4 legs
I'll remember that for next time!
Press 'q' to quit, press any other key to continue: q

C:\Users\Naafiul Hossain\Documents\Lab2\Lab1Task1\NHLab9Part6Bonous\x64\Debug\NHLab9Part6Bonous.exe
ed with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatic
le when debugging stops.
Press any key to close this window . . .
```