Experiment 6(B)

Student Name: Tanmaya Kumar Pani	**UID: 22BCS12986**
Branch: BE-CSE	**Section/Group: IOT-613B**
Semester: 5	**Date of Performance: 10/9/24**
Subject Name: Advanced Programming Lab-1	**Subject Code: 22CSP-314**

1. Title: Tree: Inorder Traversal

2. Objective:

Complete the **inOrder** function in your editor below, which has 1 parameter: a pointer to the root of a binary tree. It must print the values in the tree's inorder traversal as a single line of space-separated values.

3. Algorithm:

a) **Define Node Structure:**

- Create a Node with data, left, and right pointers. Initialize left and right to nullptr.

b) **Insert Function:**

- Insert Node into BST:
    1. If root is nullptr, create a new node with given data.
    2. If data < root->data, insert into left subtree.
    3. Otherwise, insert into right subtree.
    4. Return the updated root.

c) **Inorder Traversal Function:**

- Traverse Tree:
    1. If root is nullptr, return.
    2. Recursively traverse left subtree.
    3. Print root->data.
    4. Recursively traverse right subtree.

d) **Main Function:**

- Input and Tree Construction:
    1. Read number of nodes, n.
    2. Initialize root as nullptr.
    3. For n values, insert each into the BST.
- Traversal and Output:
    1. Perform inorder traversal and print node values.

4. Implementation/Code:

```
#include <iostream>
using namespace std;

// Structure of a tree node
struct Node {
    int data;
    Node* left;
    Node* right;

    Node(int val) {
        data = val;
        left = right = nullptr;
    }
};

// Insert value into a binary search tree
Node* insert(Node* root, int data) {
    if (root == nullptr) {
        return new Node(data);
    }

    if (data < root->data) {
        root->left = insert(root->left, data);
    } else {
        root->right = insert(root->right, data);
    }
    return root;
}

// Inorder traversal of the binary tree
void inOrder(Node* root) {
    if (root == nullptr) {
        return;
    }

    // Traverse the left subtree
    inOrder(root->left);

    // Print the value of the current node
    cout << root->data << " ";

    // Traverse the right subtree
    inOrder(root->right);
}
```

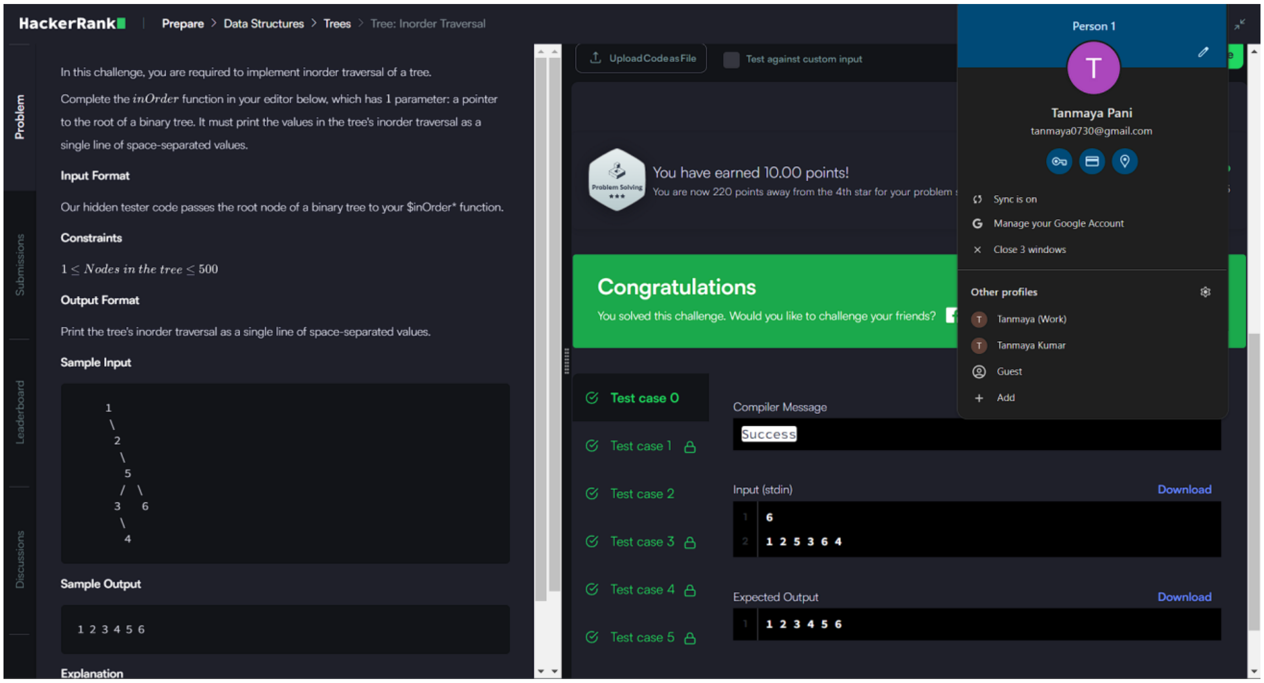
```
int main() {
    int n;
    cin >> n; // Input no. of nodes

    Node* root = nullptr;
    for (int i = 0; i < n; i++) {
        int value;
        cin >> value;
        root = insert(root, value); // Insert value into BST
    }

    // Perform inorder traversal and print result
    inOrder(root);

    return 0;
}
```

5. Output:



In this challenge, you are required to implement inorder traversal of a tree.

Complete the $inOrder$ function in your editor below, which has 1 parameter: a pointer to the root of a binary tree. It must print the values in the tree's inorder traversal as a single line of space-separated values.

Input Format

Our hidden tester code passes the root node of a binary tree to your $inOrder* function.

Constraints

$1 \leq Nodes\ in\ the\ tree \leq 500$

Output Format

Print the tree's inorder traversal as a single line of space-separated values.

Sample Input

```
     1
      \
       2
        \
         5
        /  \
       3    6
        \
         4
```

Sample Output

```
1 2 3 4 5 6
```

Explanation

You have earned 10.00 points!
You are now 220 points away from the 4th star for your problem

Congratulations
You solved this challenge. Would you like to challenge your friends?

Tanmaya Pani
tanmaya0730@gmail.com

✓ Test case 0
✓ Test case 1
✓ Test case 2
✓ Test case 3
✓ Test case 4
✓ Test case 5

Compiler Message

```
Success
```

Input (stdin) Download

```
6
1 2 5 3 6 4
```

Expected Output Download

```
1 2 3 4 5 6
```

6. Learning Outcomes:

- **Binary Search Tree (BST) Operations:** Understand how to insert nodes into a BST and maintain its properties.
- **Node Structure:** Learn to define a tree node with data, left, and right pointers in C++.
- **Recursive Functions:** Gain experience in using recursion for insertion and inorder traversal of a binary tree.
- **Inorder Traversal:** Understand the process and purpose of inorder traversal for outputting nodes in sorted order.
- **Dynamic Tree Construction:** Learn how to dynamically construct a binary tree from input and traverse it to display results.

7. Time Complexity: O(n)

8. Space Complexity: O(n)