Experiment 6

Student Name: Gaganjot Singh
UID: 22BCS14843
Branch: BE - CSE
Section/Group: 22BCS-JT-802-B
Semester: 5th
Date of Performance: 16/09/2024
Subject Name: Advance Programming
Subject Code: 22CSP - 314

Question 1.

1. Aim:

You are given a pointer to the root of a binary search tree and values to be inserted into the tree. Insert the values into their appropriate position in the binary search tree and return the root of the updated binary tree. You just have to complete the function.

2. Objective:

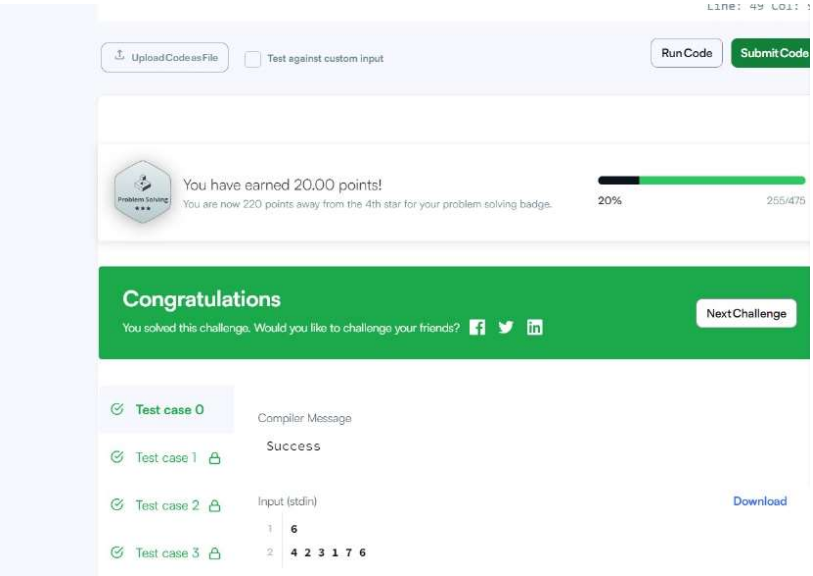
The objective of this function is to insert a new node with a specified data value into its correct position within a binary search tree (BST). If the tree is empty, it creates a new node. If the tree is not empty, it recursively traverses the left or right subtree based on the comparison between the data and the root's value, ensuring the BST property is maintained. The function returns the updated root of the tree.

3. Implementation/Code:

```
Node* insert(Node* root, int data) {
    if (root == NULL) {
        return new Node(data);
    }
    if (data < root->data) {
        root->left = insert(root->left, data);
```

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

4. Output:



5. Learning Outcomes:

- Understand how to implement the insertion operation in a binary search tree (BST) while maintaining its properties.
- Learn to use recursion for traversing and modifying tree structures.
- Recognize the significance of base cases, like checking if a node is NULL, in recursive algorithms.
- Grasp the concept of comparing node values to determine whether to insert data in the left or right subtree.
- Appreciate the importance of returning the root node to maintain the overall structure of the tree.

Question 2.

1. Aim:

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.

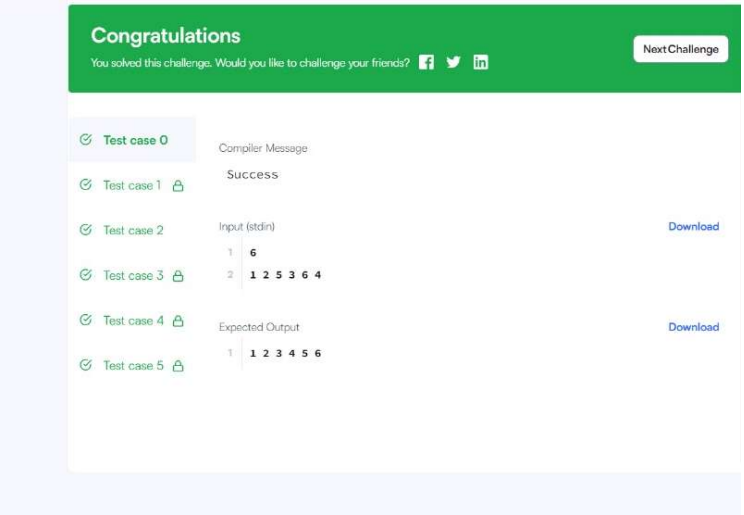
2. Objective:

The objective of this function is to perform an in-order traversal of a binary tree. It recursively visits the left subtree, prints the root node's data, and then visits the right subtree. This traversal ensures that the nodes are visited in a sorted order for binary search trees (BST).

3. Implementation/Code:

```
void inOrder(Node *root) {
    if (root == NULL) {
        return;
    }
    inOrder(root->left);
    cout << root->data << " ";
    inOrder(root->right);
}
```

4. Output:



5. Learning Outcomes:

- Understand the concept of in-order traversal in binary trees.
- Learn to apply recursion for tree traversal.
- Grasp the base case handling in recursive functions.
- Recognize the order of node visits: left subtree, root, right subtree.