Experiment 6

Student Name: Amit Kumar **UID: 22BCS16121**
Branch: CSE **Section/Group:641-A**
Semester: 6 **Date of Performance: 11/03/25**
Subject Name: AP LAB-II **Subject Code: 22CSP-351**

1. Aim:

a. To find and implement the maximum depth of Binary Tree.

b. To develop an algorithm for Binary Tree Inorder traversal.

c. Implement the concept of Symmetric Tree

2. Objective:

To implement and analyze maximum depth of Binary Tree.

To develop an algorithm for Binary Tree Inorder traversal.

To determine whether a given binary tree is **symmetric** around its center.

3. Implementation/Code:

a.
```
class Solution {
public:
    int maxDepth(TreeNode* root) {
        if (!root) return 0;
        int leftDepth = maxDepth(root->left);
        int rightDepth = maxDepth(root->right);
        return 1 + max(leftDepth, rightDepth);
    }
};
```

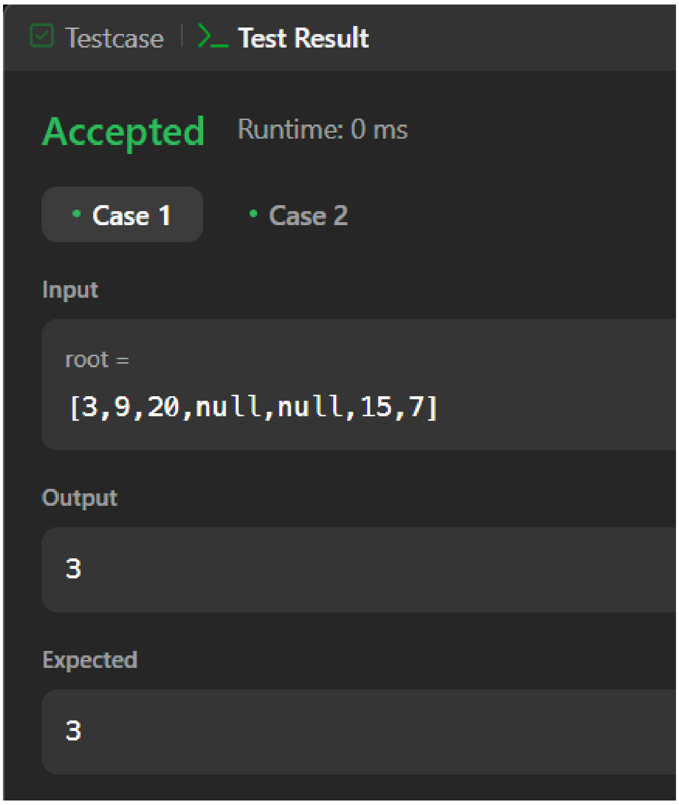
b.
```
class Solution {
public:
    vector<int> inorderTraversal(TreeNode* root) {
        vector<int>ans;
        in(root,ans);
        return ans;
    }
    void in(TreeNode* root,vector<int>&ans)
    {
        if(root==NULL)
        return;
        in(root->left,ans);
        ans.push_back(root->val);
        in(root->right,ans);
    }
};
```

c.

```
class Solution {
public:
    bool isSymmetric(TreeNode* root) {
        // If the tree is empty, it's symmetric
        if (!root) return true;

        // Helper function to check if two trees are mirror images
        return isMirror(root->left, root->right);
    }

    bool isMirror(TreeNode* left, TreeNode* right) {
        // If both are nullptr, they are mirrors
        if (!left && !right) return true;
        // If one of them is nullptr, they are not mirrors
        if (!left || !right) return false;

        // Check if the current nodes' values are equal
        // and recursively check their left and right children
        return (left->val == right->val)
            && isMirror(left->left, right->right)
            && isMirror(left->right, right->left);
    }
};
```

4. Output:

< a >



Accepted Runtime: 0 ms

Input

root =

```
[3,9,20,null,null,15,7]
```

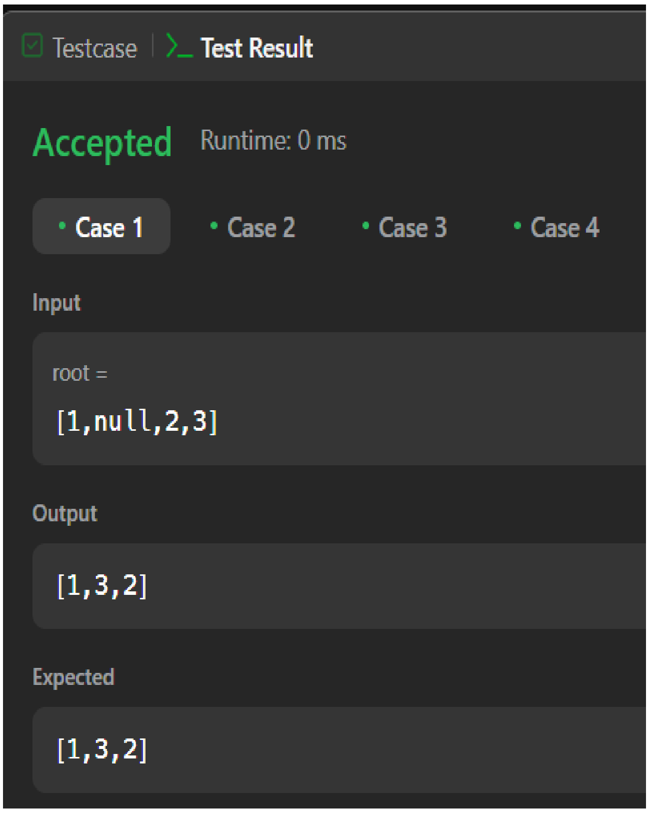
Output

```
3
```

Expected

```
3
```

< b >



Accepted Runtime: 0 ms

Input

root =

```
[1,null,2,3]
```

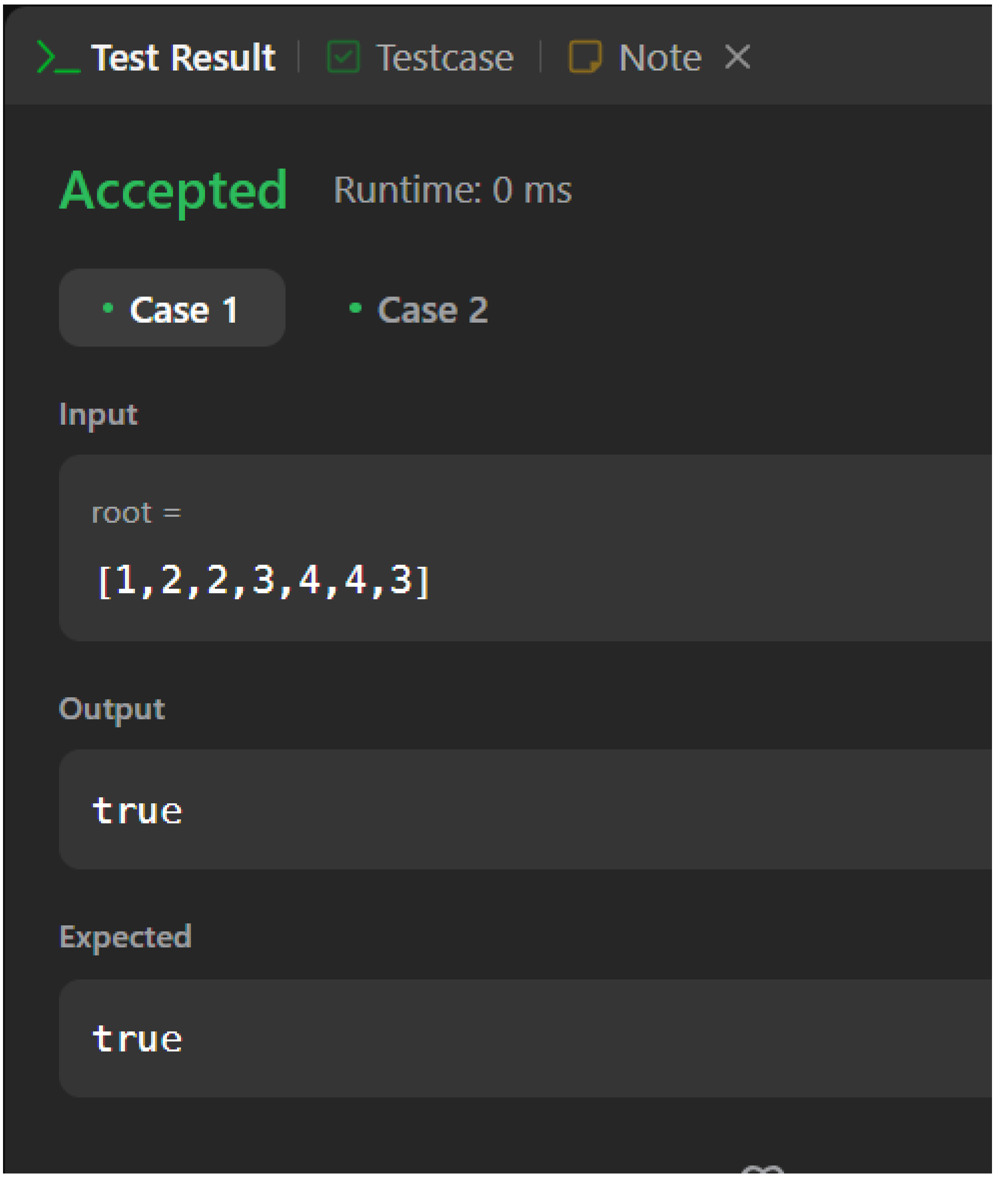
Output

```
[1,3,2]
```

Expected

```
[1,3,2]
```

< c >



Accepted Runtime: 0 ms

Input

root =
[1,2,2,3,4,4,3]

Output

```
true
```

Expected

```
true
```

4. Learning Outcome:

- Understand string manipulation techniques in C++.
- Implement efficient algorithms for detecting cyclic rotations.
- Apply mathematical approaches to solve missing number problems.
- Utilize standard library functions like accumulate and find.
- Enhance problem-solving skills through algorithm design and analysis.