Experiment 5

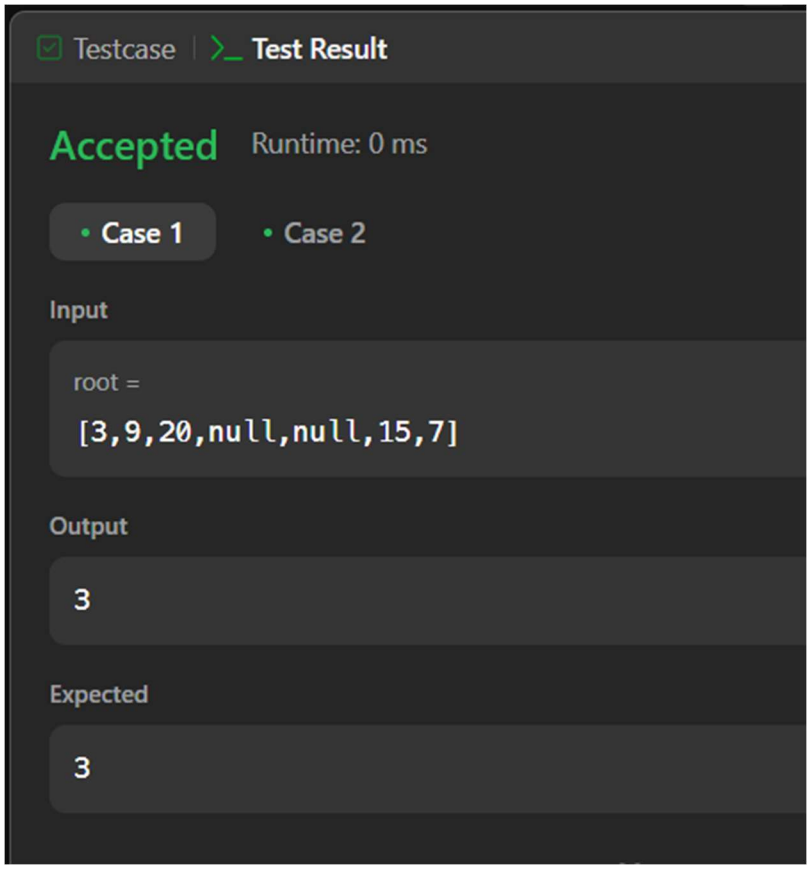
Student Name: Nikhil Niraj Singh **UID:22BCS10133**
Branch: CSE **Section/Group:22BCS_KPIT-901/B**
Semester: 6 **Date of Performance:8/1/25**
Subject Name:Advance Programming Lab **Subject Code: 22CSP-351**

1. **Aim:** To determine the maximum depth of a given binary tree by implementing an algorithm that traverses the tree and calculates the longest path from the root node to any leaf node.
2. **Objective:**
   Develop a function that, given the root of a binary tree, returns its maximum depth. The maximum depth is defined as the number of nodes along the longest path from the root node down to the farthest leaf node.

3. **Implementation/Code:**

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

4. Output:



5. Learning Outcome:

- Understand Binary Tree Structures: Comprehend the components and properties of binary trees, including nodes, root, leaves, and the concept of tree depth.

- Implement Tree Traversal Algorithms: Apply depth-first search (DFS) techniques, such as recursion, to traverse binary trees effectively.

- Calculate Maximum Depth: Develop algorithms to compute the maximum depth of a binary tree by evaluating the longest path from the root to any leaf node

1. **AIM:** To determine whether a given binary tree is symmetric around its center by implementing algorithms that verify if the tree is a mirror image of itself.
2. **Objectives:**

- Develop a function that, given the root of a binary tree, checks whether it is a mirror of itself (i.e., symmetric around its center)..

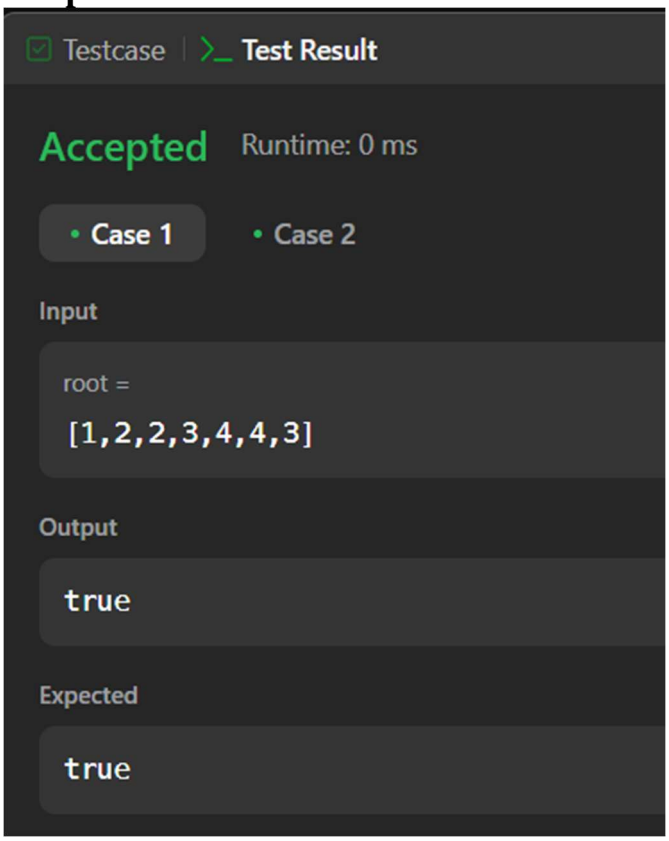
3. **Implementation code:**

```
class Solution {
 public:
  bool isSymmetric(TreeNode* root) {
    return isSymmetric(root, root);
  }

 private:
  bool isSymmetric(TreeNode* p, TreeNode* q) {
    if (!p || !q)
      return p == q;

    return p->val == q->val &&               //
           isSymmetric(p->left, q->right) &&  //
           isSymmetric(p->right, q->left);
  }
};
```

4. **Output:**



5. **Learning Outcomes:**

- Understand Binary Tree Symmetry: Comprehend the concept of tree symmetry, including how to identify mirror image structures within binary trees.
- Implement Recursive Solutions: Apply recursive techniques to traverse and compare nodes in a binary tree to determine symmetry.
- Develop Iterative Approaches: Utilize data structures such as queues or stacks to implement iterative methods for checking tree symmetry.

.