DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Experiment- 5A

Student Name:Karanvir Singh **UID:22BCS16269**

Branch:BE-CSE **Section/Group: NTPP 602-A**

Semester:6TH **Date of Performance:10/02/25**

Subject Name: AP Lab-2 **Subject Code: 22CSH-352**

1. TITLE:

Maximum Depth of Binary Tree

2. AIM:

A binary tree's maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

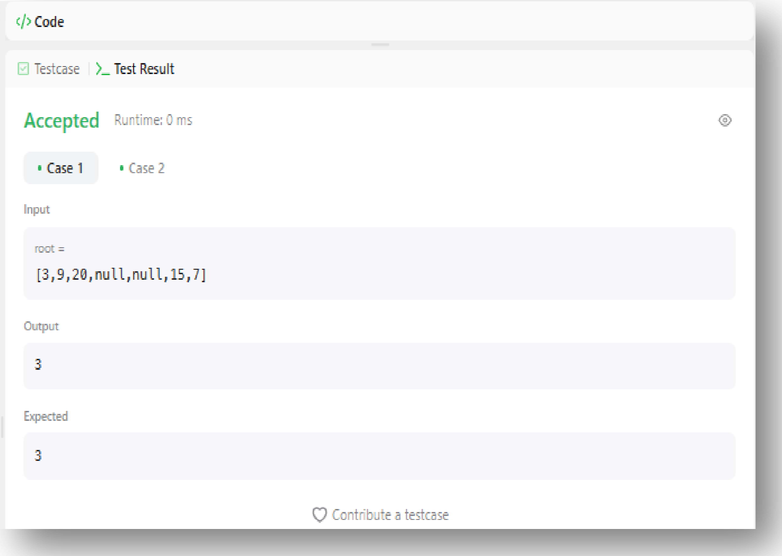
3. Algorithm

- If the node is None, return 0 (an empty tree has depth 0).
- Compute the depth of the left subtree.
- Compute the depth of the right subtree.
- The maximum depth is 1 + max(left_depth, right_depth).

Implemetation/Code

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

Output:



Accepted Runtime: 0 ms

Input

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

Output
```
3
```

Expected
```
3
```

Time Complexity : O(n)

Space Complexity : O(h)

Learning Outcomes:-

- Learn **Depth-First Search (DFS)** recursion to explore all paths in a binary tree.
- Understand **base cases** and **recursive cases** in tree structures.

Experiment-5B

Student Name: Karanvir Singh	**UID: 22BCS16269**
Branch:BE-CSE	**Section/Group: NTPP- 602(A)**
Semester:6TH	**Date of Performance:10/02/25**
Subject Name: AP Lab-2	**Subject Code: 22CSH-352**

1. TITLE:

Symmetric Tree.

2. AIM:

Given the root of a binary tree, check whether it is a mirror of itself (i.e., symmetric around itscenter).

3. Algorithm

- If the tree is empty (`root is None`), return `True`.
- Check if the **left subtree** is a mirror of the **right subtree**.
- Two subtrees are mirrors if:
- Their root values are equal.
- The **left subtree of one** matches the **right subtree of the other**.

Implemetation/Code:

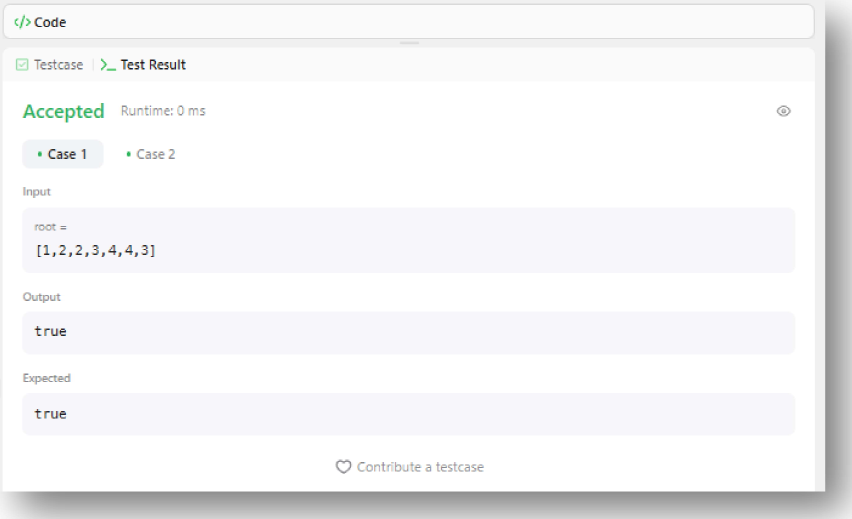
```
class Solution {
public:
bool isSymmetric(TreeNode* root) {
return isSymmetric(root, root);
}

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);
}
```

```
        };
```

Output:



Accepted Runtime: 0 ms

Input

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

Output

```
true
```

Expected

```
true
```

Time Complexity : O(N)

Space Complexity : O(h)

Learning Outcomes:-

- Learn how a tree is symmetric if its left and right subtrees are **mirror images** of each other.
- Learn how to handle **base cases** (when nodes are `None`)..