Experiment-2.1.1(Same Tree)

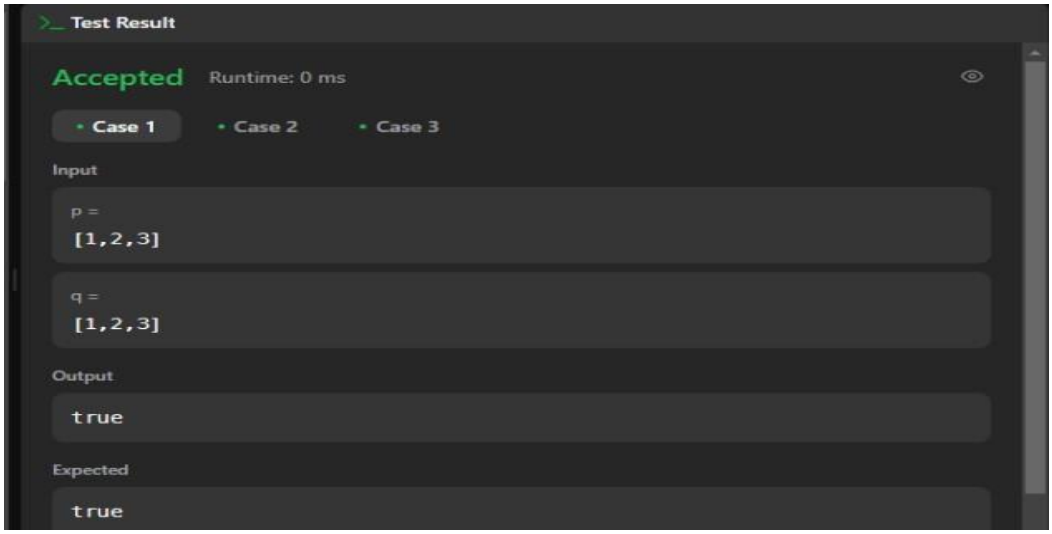
Student Name:Akash **UID:** 22BCS10329
Branch: BE-CSE **Section/Group:** IOT_638-B
Semester: 6th **Date of Performance:**21/02/25
Subject Name: AP LAB-II **Subject Code:** 22CSP-351

1. **Aim:** Given the roots of two binary trees p and q, write a function to check if they are the same or not. Two binary trees are considered the same if they are structurally identical, and the nodes have the same value.

2. **Implementation/Code:**

```
class Solution {
    public boolean isSameTree(TreeNode p, TreeNode q) {
        if (p == null && q == null) return true;
        if (p == null || q == null) return false;
        if (p.val != q.val) return false;
        return isSameTree(p.left, q.left) && isSameTree(p.right, q.right);
    }
}
```

3. **Output:**



Leetcode Link: https://leetcode.com/problems/same-tree/

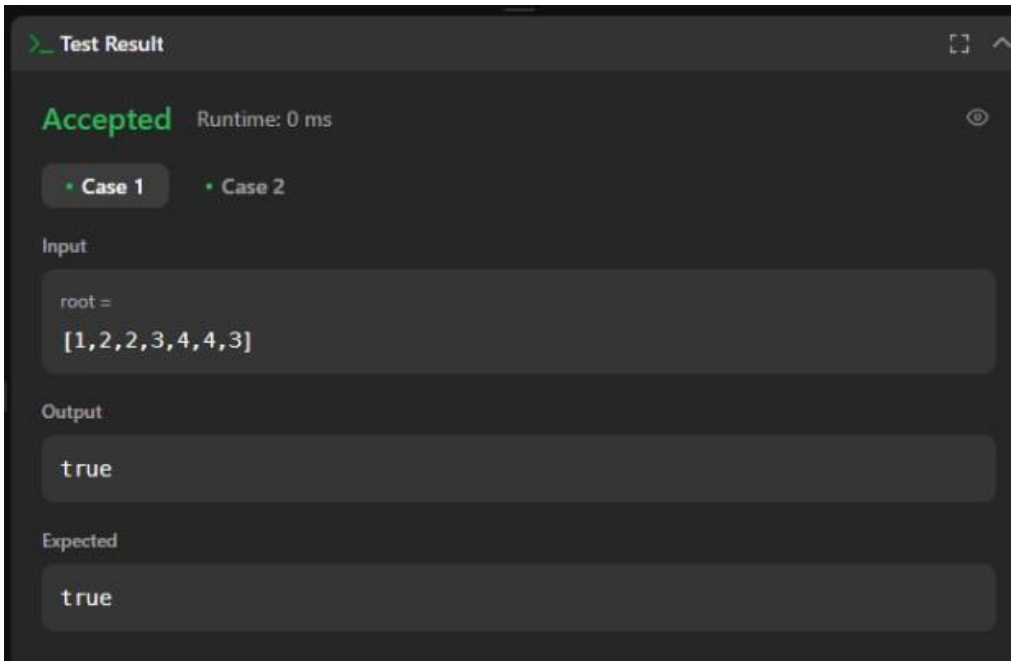
Experiment-2.1.2(Symmetric Tree)

1. **Aim:** Given the root of a binary tree, check whether it is a mirror of itself (i.e., symmetric around its center).

2. **Implementation/Code:**

```
class Solution {
    public boolean isSymmetric(TreeNode root) {
        if (root == null) return true;
        return isMirror(root.left, root.right);
    }

    private boolean isMirror(TreeNode t1, TreeNode t2) {
        if (t1 == null && t2 == null) return true;
        if (t1 == null || t2 == null) return false;
        return (t1.val == t2.val) && isMirror(t1.left, t2.right) && isMirror(t1.right, t2.left);
    }
}
```

3. **Output:**



Leetcode Link: https://leetcode.com/problems/symmetric-tree/

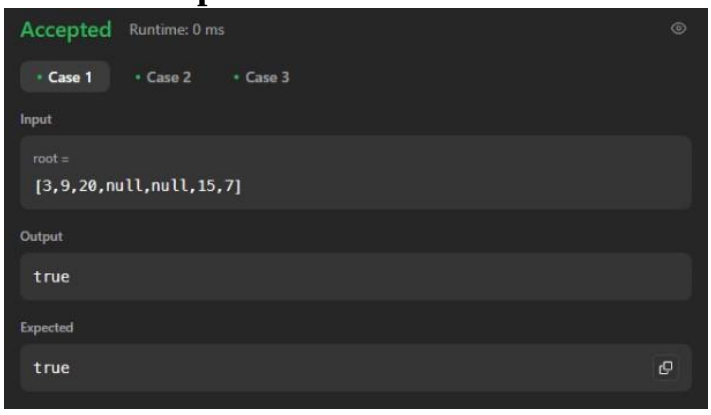
Experiment-2.1.3(Balanced Binary Tree)

1. **Aim:** Given a binary tree, determine if it is height-balanced. A binary tree is height-balanced if the difference between the heights of the left and right subtrees of any node is no more than 1.
2. **Implementation/Code:**

```
class Solution {
    public boolean isBalanced(TreeNode root) {
        return height(root) != -1;
    }

    private int height(TreeNode node) {
        if (node == null) return 0;

        int leftHeight = height(node.left);
        if (leftHeight == -1) return -1;

        int rightHeight = height(node.right);
        if (rightHeight == -1) return -1;

        if (Math.abs(leftHeight - rightHeight) > 1) return -1;

        return Math.max(leftHeight, rightHeight) + 1;
    }
        }
```

Leetcode link: https://leetcode.com/problems/balanced-binary-tree/

3. **Output:**



Experiment-2.1.4(Path Sum)

1. **Aim:** Given the root of a binary tree and an integer targetSum, return true if the tree has a root-to-leaf path such that adding up all the values along the path equals targetSum.

2. **Implementation/Code:**

```
class Solution {
    public int countNodes(TreeNode root) {
        if (root == null) return 0;

        int leftDepth = getDepth(root.left);
        int rightDepth = getDepth(root.right);

        if (leftDepth == rightDepth) {
            return (1 << leftDepth) + countNodes(root.right);
        } else {
            return (1 << rightDepth) + countNodes(root.left);
        }
    }

    private int getDepth(TreeNode node) {
        int depth = 0;
        while (node != null) {
            depth++;
            node = node.left;
        }
        return depth;
    }
}
```

Leetcode link: https://leetcode.com/problems/count-complete-tree-nodes/

3. Output:

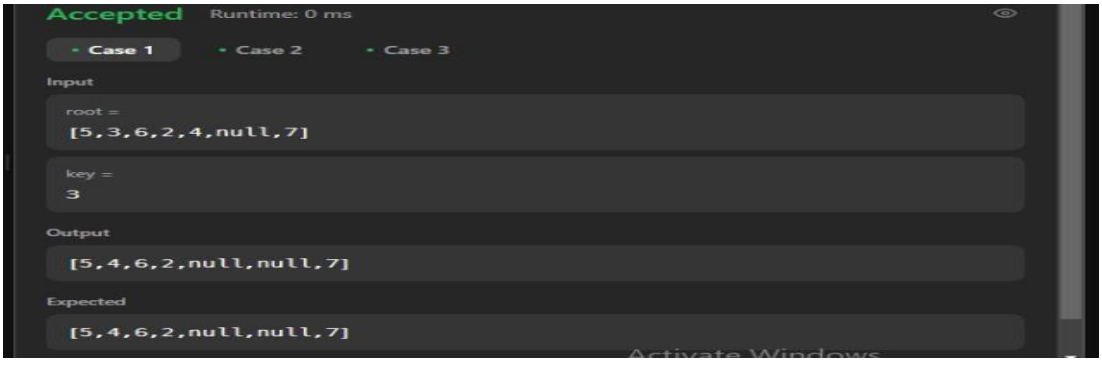

Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
root =
[1,2,3,4,5,6]
Output
6
Expected
6

Experiment-2.1.5(Delete Node in a BST)

1. **Aim:** Given the root of a BST and a key, delete the node with the given key in the BST.
2. **Implementation/Code:**

```
class Solution {
    public TreeNode deleteNode(TreeNode root, int key) {
        if (root == null) return null;

        if (key < root.val) {
            root.left = deleteNode(root.left, key);
        } else if (key > root.val) {
            root.right = deleteNode(root.right, key);
        } else {
            if (root.left == null) return root.right;
            if (root.right == null) return root.left;

            TreeNode minNode = getMin(root.right);
            root.val = minNode.val;
            root.right = deleteNode(root.right, minNode.val);
        }
        return root;
    }

    private TreeNode getMin(TreeNode node) {
        while (node.left != null) {
            node = node.left;
        }
        return node;
    }
}
```

3. **Output:**



Leetcode Link: https://leetcode.com/problems/delete-node-in-a-bst/

Experiment-2.1.6(Count Complete Tree Nodes)

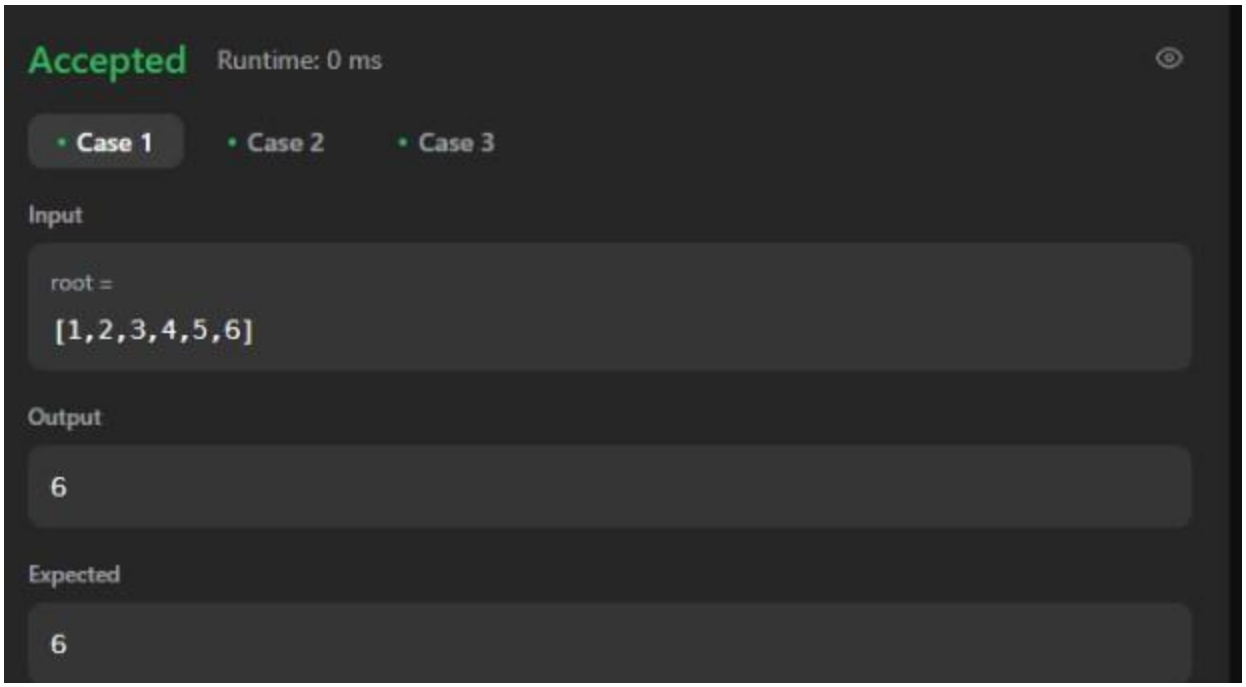
1. **Aim:** Given the root of a binary tree, check whether it is a mirror of itself (i.e., symmetric around its center).

2. **Implementation/Code:**

```
class Solution {
    public int countNodes(TreeNode root) {
        if (root == null) return 0;

        int leftDepth = getDepth(root.left);
        int rightDepth = getDepth(root.right);

        if (leftDepth == rightDepth) {
            return (1 << leftDepth) + countNodes(root.right);
        } else {
            return (1 << rightDepth) + countNodes(root.left);
        }
    }

    private int getDepth(TreeNode node) {
        int depth = 0;
        while (node != null) {
            depth++;
            node = node.left;
        }
        return depth;
    }
}
```

Leetcode Link: **https://leetcode.com/problems/count-complete-tree-nodes/**

3. Output:


Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
root =
[1,2,3,4,5,6]
Output
6
Expected
6

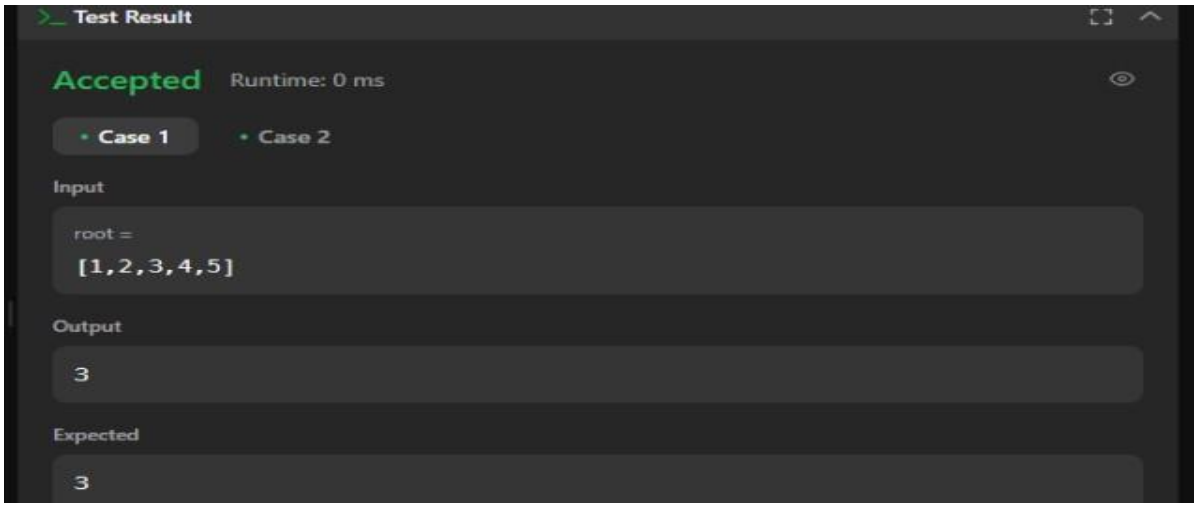
Experiment-2.1.7(Diameter of Binary Tree)

1. **Aim:** Given the root of a binary tree, return the length of the diameter of the tree. The diameter of a binary tree is the length of the longest path between any two nodes in a tree.

2. **Implementation/Code:**

```
class Solution {
    private int diameter = 0;

    public int diameterOfBinaryTree(TreeNode root) {
        depth(root);
        return diameter;
    }

    private int depth(TreeNode node) {
        if (node == null) return 0;
        int leftDepth = depth(node.left);
        int rightDepth = depth(node.right);
        diameter = Math.max(diameter, leftDepth + rightDepth);
        return Math.max(leftDepth, rightDepth) + 1;
```

3. **Output:**



Leetcode Link: https://leetcode.com/problems/diameter-of-binary-tree/