

## Experiment No. - 5

**Student Name: Harshit Raj**

**Branch: BE-CSE**

**Semester: 6<sup>th</sup>**

**Subject Name: Competitive coding - II**

**UID: 20BCS9266**

**Section/Group: 20BCS-608/A**

**Date of Performance: 05/04/2023**

**Subject Code: 20CSP-351**

### Aim/Overview of the practical

#### Q.1 Balance Binary Tree.

<https://leetcode.com/problems/balanced-binary-tree/>

### Apparatus / Simulator Used:

- Windows 7 or above
- Google Chrome

### Objective:

- To understand the concept of Tree
- To implement the concept of Balance Binary Tree.

### Code:

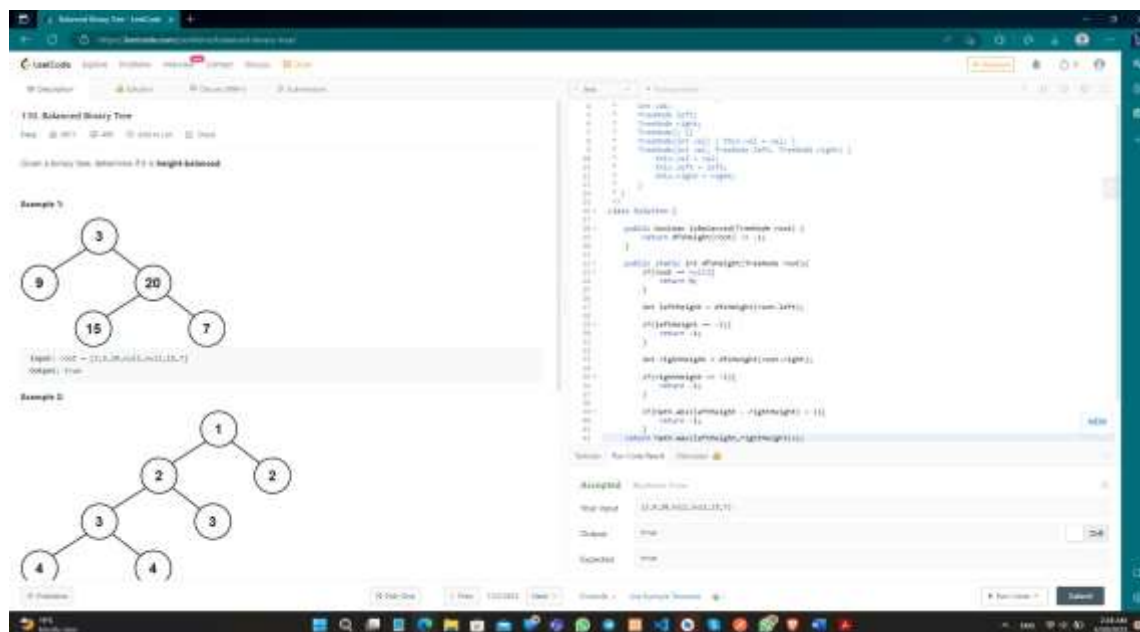
```
class Solution {  
  
    public boolean isBalanced(TreeNode root) {  
        return dfsHeight(root) != -1;  
    }  
  
    public static int dfsHeight(TreeNode root){  
        if(root == null){  
return 0;  
        }  
  
        int leftHeight = dfsHeight(root.left);  
  
        if(leftHeight == -1){  
            return -1;  
        }  
  
        int rightHeight = dfsHeight(root.right);  
  
        if(rightHeight == -1){  
            return -1;  
        }  
    }  
}
```

```

        if(Math.abs(leftHeight - rightHeight) > 1){
            return -1;
        }
        return Math.max(leftHeight,rightHeight)+1;
    } }

```

### Result/Output/Writing Summary:



### Aim/Overview of the practical:

#### Q.2 Path Sum

<https://leetcode.com/problems/path-sum/>

### Apparatus / Simulator Used:

- Windows 7 or above
- Google Chrome

### Objective:

□ To understand the concept of Tree traversal. □ To implement the concept of calculate the path sum.

### Code:

```

class Solution {
    public boolean hasPathSum(TreeNode root, int targetSum) {
        if (root == null) {
            return false;

```

```

    }
    if (root.val == targetSum && root.left == null && root.right == null) {
return true;
    }
    return hasPathSum(root.left, targetSum - root.val) || hasPathSum(root.right, targetSum - root.val);
}
}

```

### Result/Output/Writing Summary:

The screenshot displays the LeetCode interface for problem 112, "Path Sum". On the left, the problem description states: "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. A leaf is a node with no children." Example 1 shows a tree with root 5, left child 4, right child 8, and further children leading to a path sum of 22. Example 2 shows a single node 1. On the right, a Java solution is provided, implementing a recursive function `hasPathSum` that checks if a root-to-leaf path exists with the given target sum.

### Learning outcomes (What I have learnt):

- Learned the concept of Balanced Binary Tree.
- Learnt about Tree and Path Sum.