Experiment – 2.1

Student Name: Subhadip Patra **UID: 20BCS2543**

Branch: CSE **Section/Group: 707/B**

Subject Name: Competitive Coding II **Subject Code: 20CSP-351**

1. Aim: Balanced Binary Tree

2. Objective:

Given a binary tree, determine if it is height-balanced.

Example 1:

Input: root = [3,9,20,null,null,15,7]

Output: true

Example 2:

Input: root = [1,2,2,3,3,null,null,4,4]

Output: false

Example 3:

Input: root = []

Output: true

Constraints:

The number of nodes in the tree is in the range [0, 5000].

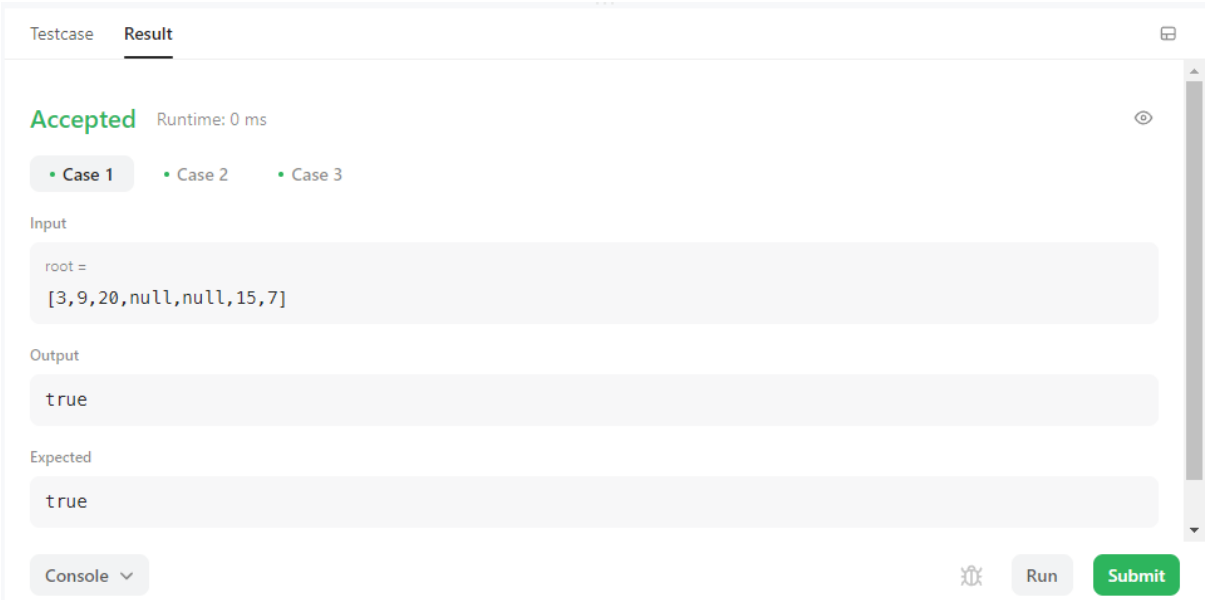
-104 <= Node.val <= 104

3. Code:

```
class Solution {
    public int ht(TreeNode root)
    {
        if(root==null)
            return -1;
        return Math.max(ht(root.left),ht(root.right))+1;
    }
    public boolean isBalanced(TreeNode root)
    {
     if(root==null)
         return true;
     boolean l=isBalanced(root.left);
      boolean r=isBalanced(root.right);

        int lht=ht(root.left);
        int rht=ht(root.right);
        if(Math.abs(lht-rht)>=2)
            return false;
        return l&&r;


    }
}
```

4. Output:

```
class Solution {
    public int ht(TreeNode root)
    {
        if(root==null)
            return -1;
        return Math.max(ht(root.left),ht(root.right))+1;
    }
    public boolean isBalanced(TreeNode root)
    {
     if(root==null)
         return true;
     boolean l=isBalanced(root.left);
      boolean r=isBalanced(root.right);

        int lht=ht(root.left);
        int rht=ht(root.right);
        if(Math.abs(lht-rht)>=2)
            return false;
        return l&&r;


    }
}
```



5. Aim: Path Sum

6. Objective:

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:
Input: root = [5,4,8,11,null,13,4,7,2,null,null,null,1], targetSum = 22
Output: true
Explanation: The root-to-leaf path with the target sum is shown.

Example 2:
Input: root = [1,2,3], targetSum = 5
Output: false
Explanation: There two root-to-leaf paths in the tree:
(1 --> 2): The sum is 3.
(1 --> 3): The sum is 4.
There is no root-to-leaf path with sum = 5.

Example 3:
Input: root = [], targetSum = 0
Output: false
Explanation: Since the tree is empty, there are no root-to-leaf paths.

Constraints:
The number of nodes in the tree is in the range [0, 5000].
-1000 <= Node.val <= 1000
-1000 <= targetSum <= 1000

7. **Code:**

```
class Solution {
    public boolean rootToLeafPathSum(TreeNode root, int targetSum, int
sum){
        if(root == null)
            return false;
        if(root.left == null && root.right == null){
            sum = sum + root.val;
            if(sum == targetSum)
                return true;
        }
        return rootToLeafPathSum(root.left, targetSum, sum + root.val) ||
rootToLeafPathSum(root.right, targetSum, sum + root.val);
    }
    public boolean hasPathSum(TreeNode root, int targetSum) {
        int sum = 0;
        return rootToLeafPathSum(root, targetSum, sum);
    }
}
```

8. Output:

```java
class Solution {
    public boolean rootToLeafPathSum(TreeNode root, int targetSum, int sum){
        if(root == null)
            return false;
        if(root.left == null && root.right == null){
            sum = sum + root.val;
            if(sum == targetSum)
                return true;
        }
        return rootToLeafPathSum(root.left, targetSum, sum + root.val) || rootToLeafPathSum(root.right, targetSum, sum + root.val);
    }
    public boolean hasPathSum(TreeNode root, int targetSum) {
        int sum = 0;
        return rootToLeafPathSum(root, targetSum, sum);
    }
}
```