## DAY 13:

## ASSIGNMENT 1:

### Q) Balanced Binary Tree Check

Write a function to check if a given binary tree is balanced. A balanced tree is one where the height of two subtrees of any node never differs by more than one.

### ANSWER:

```
package day13;

        class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int x) {
        val = x;
    }
}

public class BalancedBinaryTreeChecker {

    public boolean isBalanced(TreeNode root) {
        return checkHeight(root) != -1;
    }

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

        int leftHeight = checkHeight(node.left);
        if (leftHeight == -1) {
```

```
        return -1; // Not balanced
    }

    int rightHeight = checkHeight(node.right);
    if (rightHeight == -1) {
        return -1; // Not balanced
    }

    if (Math.abs(leftHeight - rightHeight) > 1) {
        return -1; // Not balanced
    }

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

public static void main(String[] args) {
    // Example usage:
    // Constructing a simple balanced binary tree
    //       1
    //      / \
    //     2   3
    //    / \
    //   4   5
    TreeNode root = new TreeNode(1);
    root.left = new TreeNode(2);
    root.right = new TreeNode(3);
    root.left.left = new TreeNode(4);
    root.left.right = new TreeNode(5);

    BalancedBinaryTreeChecker treeChecker = new BalancedBinaryTreeChecker();
    System.out.println(treeChecker.isBalanced(root));  // Output: true
```

```
}
```

## Explanation:

### 1. TreeNode Class: Defines the structure of a node in the binary tree, with an integer value and references to left and right children.

### 2. BalancedBinaryTreeChecker Class : Contains the method isBalanced and a helper method checkHeight.

### 3. isBalanced Method:

- Calls the helper method checkHeight and checks if its return value is not -1. If checkHeight returns -1, it means the tree is not balanced.

### 4. checkHeight Method:

- If the node is null, it returns a height of 0 (base case).

- Recursively computes the height of the left subtree. If the left subtree is not balanced (i.e., checkHeight returns -1), it immediately returns -1.

- Similarly, it computes the height of the right subtree and checks if it's balanced.

- If the height difference between the left and right subtrees is more than 1, it returns -1, indicating the subtree rooted at the current node is not balanced.

- If balanced, it returns the height of the current node, which is one more than the height of its tallest subtree.

### 5. Main Method: Demonstrates how to use the BalancedBinaryTreeChecker class to check if a sample tree is balanced.

This implementation follows the same O(n) time complexity, ensuring each node is visited only once. This method uses a sentinel value (-1) to propagate the imbalance status up the recursive call stack efficiently.