Practical -15

Write a Program for Building a Function ISVALID to VALIDATE BST

Algorithm to Validate a BST

1. **Define a Recursive Helper Function**:
    - Create a function **isValidBSTHelper(node, min, max)** that takes a node and the valid range defined by **min** and **max**.
2. **Base Case**:
    - If the **node** is **null**, return **true** (an empty tree is a valid BST).
3. **Check Current Node**:
    - If the value of the **node** is less than or equal to **min** or greater than or equal to **max**, return **false** (the current node violates the BST property).
4. **Recur for Subtrees**:
    - Recursively call **isValidBSTHelper** for the left child with an updated maximum value (**node.val** as the new max).
    - Recursively call **isValidBSTHelper** for the right child with an updated minimum value (**node.val** as the new min).
5. **Combine Results**:
    - Return the logical AND of the results from the left and right subtree checks.
6. **Initial Call**:
    - Call the helper function from the main function with the root node and the initial range of **Long.MIN_VALUE** to **Long.MAX_VALUE**.

Code:-

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

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

public class ValidateBST {

    public boolean isValidBST(TreeNode root) {
        return isValidBSTHelper(root, Long.MIN_VALUE, Long.MAX_VALUE);
    }

    private boolean isValidBSTHelper(TreeNode node, long min, long max) {
        // Base case: an empty node is a valid BST
        if (node == null) {
            return true;
        }

        // Check if the current node's value is within the valid range
        if (node.val <= min || node.val >= max) {
            return false;
        }

        // Recursively check the left and right subtrees
        return isValidBSTHelper(node.left, min, node.val) &&
               isValidBSTHelper(node.right, node.val, max);
    }

    public static void main(String[] args) {
        // Example usage:
        TreeNode root = new TreeNode(2);
        root.left = new TreeNode(1);
        root.right = new TreeNode(3);

        ValidateBST validator = new ValidateBST();
        System.out.println(validator.isValidBST(root)); // Output: true

        // Example of an invalid BST
        TreeNode invalidRoot = new TreeNode(5);
        invalidRoot.left = new TreeNode(1);
        invalidRoot.right = new TreeNode(4);
        invalidRoot.right.left = new TreeNode(3);
        invalidRoot.right.right = new TreeNode(6);

        System.out.println(validator.isValidBST(invalidRoot)); // Output: false
    }
```

```
}
```

```
true
false
```