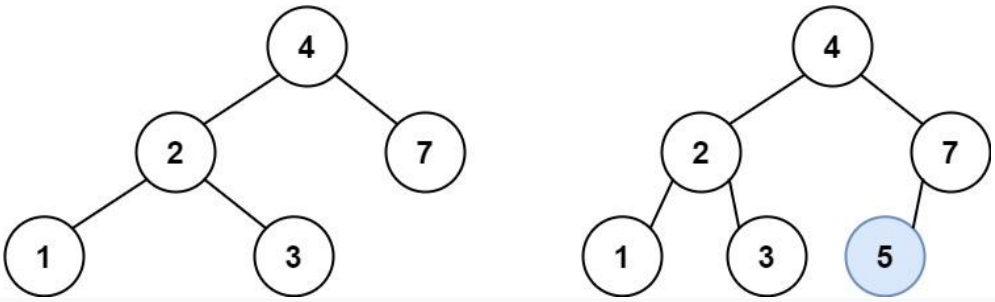
Insert into a Binary Search Tree

You are given the `root` node of a binary search tree (BST) and a `value` to insert into the tree. Return *the root node of the BST after the insertion*. It is **guaranteed** that the new value does not exist in the original BST.

Notice that there may exist multiple valid ways for the insertion, as long as the tree remains a BST after insertion. You can return **any of them**.

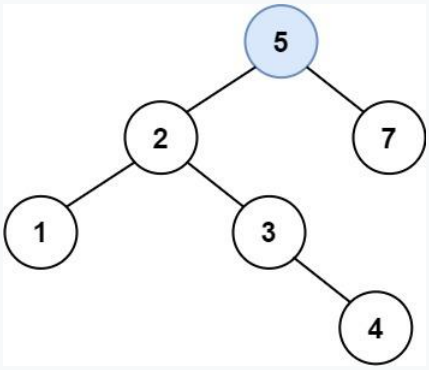
Example 1:




```
Input: root = [4,2,7,1,3], val = 5

Output: [4,2,7,1,3,5]

Explanation: Another accepted tree is:
```



Example 2:

```
Input: root = [40,20,60,10,30,50,70], val = 25

Output: [40,20,60,10,30,50,70,null,null,25]
```

Example 3:

```
Input: root = [4,2,7,1,3,null,null,null,null,null,null], val = 5

Output: [4,2,7,1,3,5]
```

Constraints:

- The number of nodes in the tree will be in the range `[0, 10⁴]`.
- `-10⁸ <= Node.val <= 10⁸`
- All the values `Node.val` are **unique**.
- `-10⁸ <= val <= 10⁸`
- It's **guaranteed** that `val` does not exist in the original BST.

```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     public int val;
 *     public TreeNode left;
 *     public TreeNode right;
 *     public TreeNode(int val=0, TreeNode left=null, TreeNode right=null) {
 *         this.val = val;
 *         this.left = left;
 *         this.right = right;
 *     }
 * }
 */

public class Solution {
    public TreeNode InsertIntoBST(TreeNode root, int val) {
        if(root == null)
        {
            root = new TreeNode(val);
        }
        else
        {
            Insert(root, val);
        }
        return root;
    }

    void Insert(TreeNode root, int val)
    {
        if(root == null)
        {
            return;
        }

        if(root.val > val)
        {
            if(root.left == null)
            {
                root.left = new TreeNode(val);
            }
            else
            {
                Insert(root.left, val);
            }
        }
        else
        {
            if(root.right == null)
            {
                root.right = new TreeNode(val);
            }
            else
            {
                Insert(root.right, val);
            }
        }
    }
}
```