Homework(17/06/2025)

Mirror Tree – Already Done.

Q1- Write a java program to take a unsorted array as input & DEFINE TreeNode & construct a BST & print all it's element in level order.

Java Program-

```
class TreeNode {
    int data;
    TreeNode left, right;

    TreeNode(int val) {
        this.data = val;
        left = right = null;
    }
}

public class BSTLevelOrder {

    public static TreeNode insert(TreeNode root, int val) {
        if (root == null)
            return new TreeNode(val);

        if (val < root.data)
            root.left = insert(root.left, val);
        else
            root.right = insert(root.right, val);
```

```
    return root;
}

public static void levelOrder(TreeNode root) {
    if (root == null) return;

    Queue<TreeNode> queue = new LinkedList<>();
    queue.add(root);

    while (!queue.isEmpty()) {
        int levelSize = queue.size();
        for (int i = 0; i < levelSize; i++) {
            TreeNode current = queue.poll();
            System.out.print(current.data + " ");

            if (current.left != null) queue.add(current.left);
            if (current.right != null) queue.add(current.right);
        }
        System.out.println();
    }
}

public static void main(String[] args) {
    int[] arr = {10, 5, 20, 3, 7, 15, 25};
```

```
        TreeNode root = null;
        for (int val : arr) {
            root = insert(root, val);
        }

        System.out.println("Level Order Traversal of BST:");
        levelOrder(root);
    }
}
```

Q2-Write a Program in java to Define TreeNode , Construct a Binary Tree & check whether it is a valid BST or Not .

Java Program-

```
class TreeNode {
    int data;
    TreeNode left, right;

    TreeNode(int val) {
        this.data = val;
        left = right = null;
    }
}

public class CheckValidBST {
    public static boolean isValidBST(TreeNode root) {
        return isBSTUtil(root, Long.MIN_VALUE, Long.MAX_VALUE);
    }
```

```
    private static boolean isBSTUtil(TreeNode node, long min, long max) {
        if (node == null) return true;

        if (node.data <= min || node.data >= max)
            return false;

        return isBSTUtil(node.left, min, node.data) &&
               isBSTUtil(node.right, node.data, max);
    }

    public static void main(String[] args) {

        TreeNode root = new TreeNode(10);
        root.left = new TreeNode(5);
        root.right = new TreeNode(15);
        root.right.left = new TreeNode(12);
        root.right.right = new TreeNode(20);

        if (isValidBST(root)) {
            System.out.println("The Binary Tree is a VALID BST.");
        } else {
            System.out.println("The Binary Tree is NOT a valid BST.");
        }
    }
}
```