Experiment 6

Student Name: Arfan Mohd **UID: 22BCS12329**
Branch: CSE **Section/Group: NTPP 603B**
Semester: 6 **Date of Performance: 18/03/25**
Subject Name: AP Lab 2 **Subject Code:22CSP-351**

1. **Aim**:
   A. Maximum Depth of Binary Tree
   B. Validate Binary Search Tree
   C. Binary Tree Level Order Traversal
   D. Kth Smallest Element in a BST

2. **Code:**

A. *104.java*

```
class Solution {
    public int maxDepth(TreeNode root) {
        if(root==null) {
        return 0; // Base case: if the tree is empty, return 0
    }

    // Recursively get the depth of the left and right subtrees
    int leftDepth=maxDepth(root.left);
    int rightDepth=maxDepth(root.right);

    // Return the maximum depth of the left and right subtrees, plus 1 for the current node
    return 1+Math.max(leftDepth, rightDepth);
    }
}
```

B. *98.java*

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

    private boolean isValidBSTHelper(TreeNode node, long min, long max) {
        if (node == null) {
```

```
        return true;
    }
    if (node.val <= min || node.val >= max) {
        return false;
    }
    return isValidBSTHelper(node.left, min, node.val) && isValidBSTHelper(node.right,
node.val, max);
  }
}
```

C. *102.java*

```
class Solution {
  public List<List<Integer>> levelOrder(TreeNode root) {
    List<List<Integer>> result = new ArrayList<>();
    if (root == null) {
      return result;
    }

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

    while (!queue.isEmpty()) {
      int levelSize = queue.size();
      List<Integer> level = new ArrayList<>();

      for (int i = 0; i < levelSize; i++) {
        TreeNode node = queue.poll();
        level.add(node.val);

        if (node.left != null) {
          queue.offer(node.left);
        }
        if (node.right != null) {
          queue.offer(node.right);
        }
      }

      result.add(level);
    }

    return result;
```

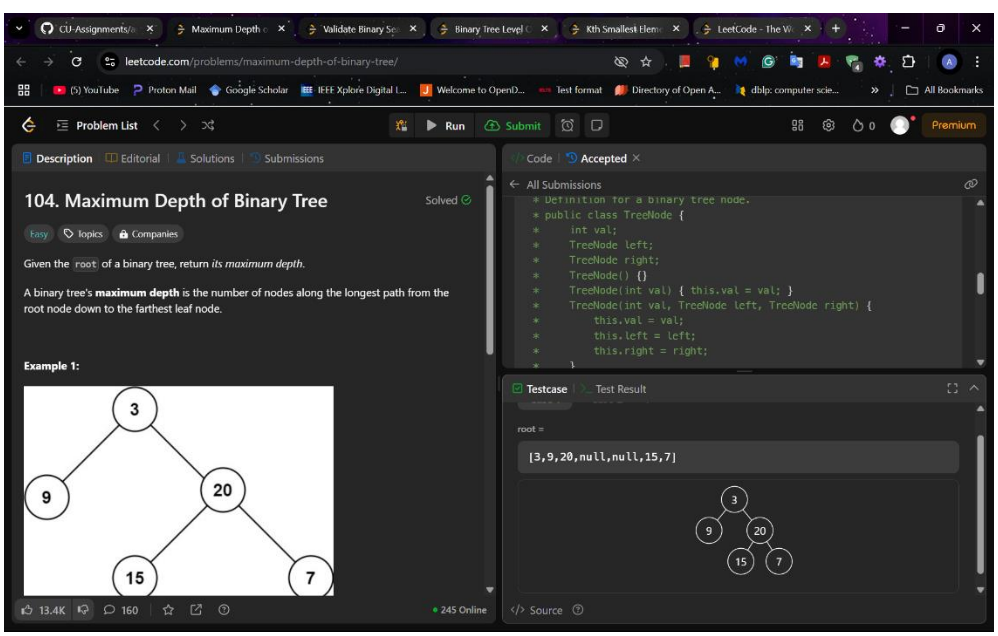
```
    }
}
```

C. *230.java*

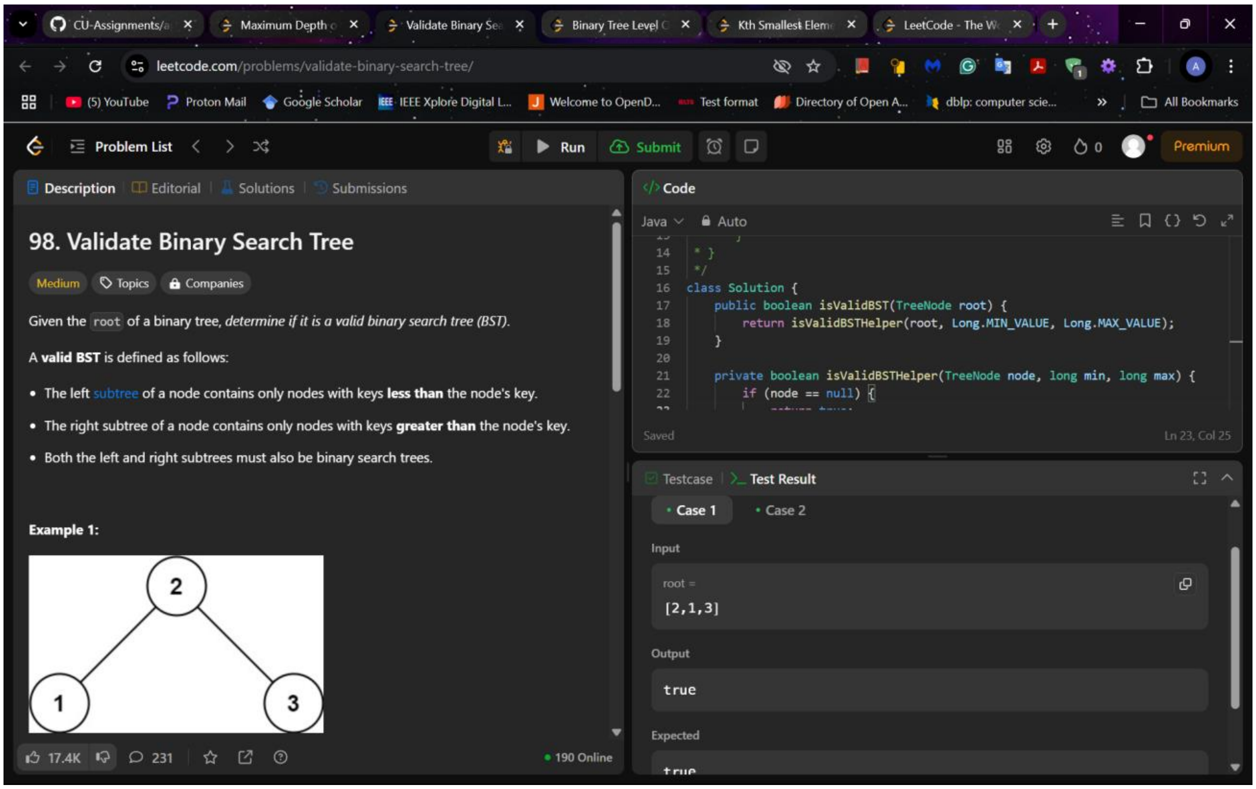
```
class Solution {
    public int kthSmallest(TreeNode root, int k) {
        // Inorder traversal will give elements in ascending order in a BST
        List<Integer> inorderList = new ArrayList<>();
        inorderTraversal(root, inorderList);
        return inorderList.get(k - 1); // k is 1-based index
    }

    // Helper method to perform inorder traversal
    private void inorderTraversal(TreeNode node, List<Integer> inorderList) {
        if (node == null) {
            return;
        }
        inorderTraversal(node.left, inorderList);  // Visit left subtree
        inorderList.add(node.val);                  // Visit node
        inorderTraversal(node.right, inorderList); // Visit right subtree
    }
}
```

3. Output:

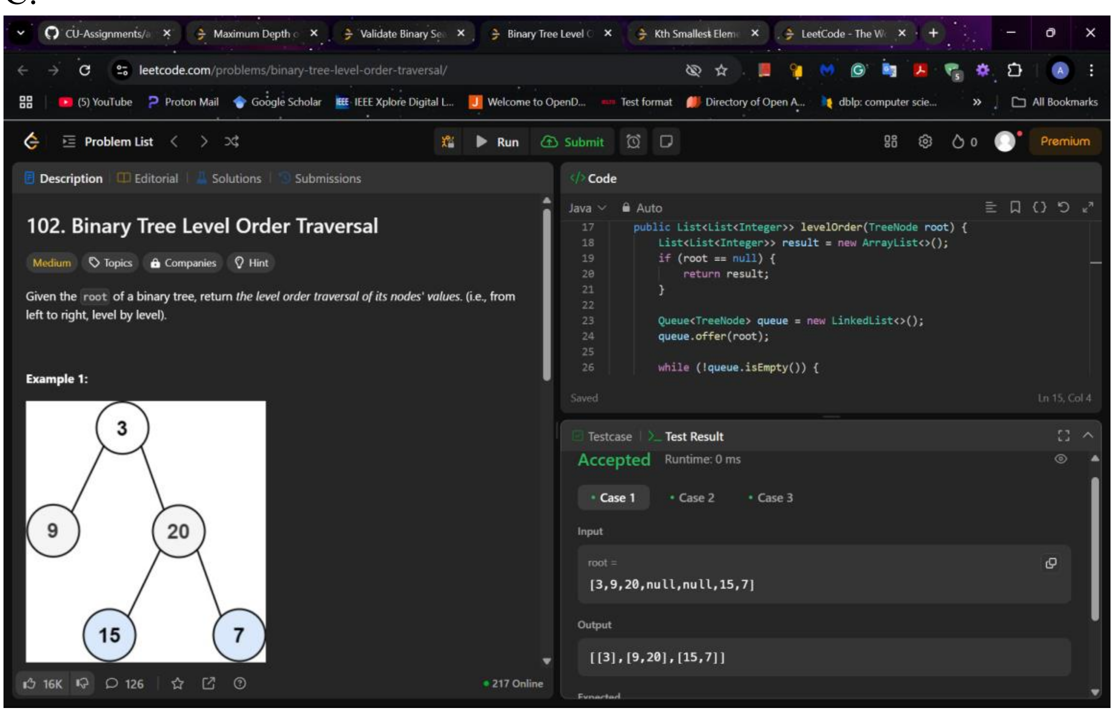
A.



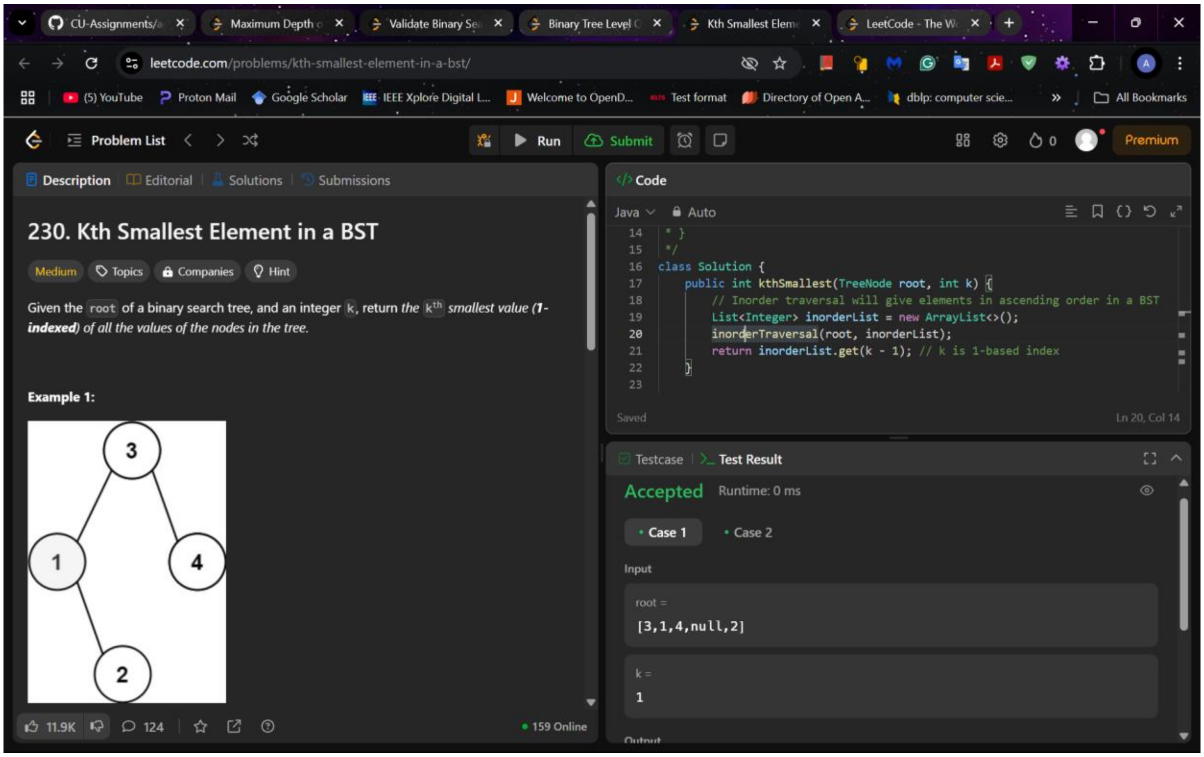
B.


leetcode.com/problems/validate-binary-search-tree/
Description
Editorial
Solutions
Submissions
98. Validate Binary Search Tree
Medium
Topics
Companies
Given the root of a binary tree, determine if it is a valid binary search tree (BST).
A valid BST is defined as follows:
The left subtree of a node contains only nodes with keys less than the node's key.
The right subtree of a node contains only nodes with keys greater than the node's key.
Both the left and right subtrees must also be binary search trees.
Example 1:
2
1
3
Code
Java
class Solution {
public boolean isValidBST(TreeNode root) {
return isValidBSTHelper(root, Long.MIN_VALUE, Long.MAX_VALUE);
}
private boolean isValidBSTHelper(TreeNode node, long min, long max) {
if (node == null) {
Testcase
Test Result
Case 1
Case 2
Input
root =
[2,1,3]
Output
true
Expected

C.


leetcode.com/problems/binary-tree-level-order-traversal/
Description
Editorial
Solutions
Submissions
102. Binary Tree Level Order Traversal
Medium
Topics
Companies
Hint
Given the root of a binary tree, return the level order traversal of its nodes' values. (i.e., from left to right, level by level).
Example 1:
3
9
20
15
7
Code
Java
public List<List<Integer>> levelOrder(TreeNode root) {
List<List<Integer>> result = new ArrayList<>();
if (root == null) {
return result;
}
Queue<TreeNode> queue = new LinkedList<>();
queue.offer(root);
while (!queue.isEmpty()) {
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
root =
[3,9,20,null,null,15,7]
Output
[[3],[9,20],[15,7]]

D.


leetcode.com/problems/kth-smallest-element-in-a-bst/
Problem List
Run
Submit
Premium
Description
Editorial
Solutions
Submissions
230. Kth Smallest Element in a BST
Medium
Topics
Companies
Hint
Given the root of a binary search tree, and an integer k, return the kth smallest value (1-indexed) of all the values of the nodes in the tree.
Example 1:
3
1
4
2
11.9K
124
159 Online
Code
Java
Auto
* }
*/
class Solution {
public int kthSmallest(TreeNode root, int k) {
// Inorder traversal will give elements in ascending order in a BST
List<Integer> inorderList = new ArrayList<>();
inorderTraversal(root, inorderList);
return inorderList.get(k - 1); // k is 1-based index
}
Saved
Ln 20, Col 14
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
root =
[3,1,4,null,2]
k =
1
Output