Experiment 6

Student Name: Dhruv Sumbria **UID:**22BET10262
Branch: BE -IT **Section/Group:**22BET/IOT/702/B
Semester: 6th **Date of Performance:28**/02/2025
Subject Name: Advanced Programming Lab-2 **Subject Code:** 22ITP-351

1. Aim 1 : Maximum Depth of Binary Tree

Given the root of a binary tree, return its maximum depth.

A binary tree's maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

2. Validate Binary Search Tree :

Given the root of a binary tree, determine if it is a valid binary search tree (BST).
A valid BST is defined as follows:

- 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.

3. Symmetric Tree:

Given the root of a binary tree, check whether it is a mirror of itself (i.e., symmetric around its center).

4. Binary Tree Level Order Traversal:

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).

5. Convert Sorted Array to Binary Search Tree:

Given an integer array nums where the elements are sorted in ascending order, convert it to a height-balanced binary search tree.

6. Binary Tree Inorder Traversal:

Given the root of a binary tree, return the inorder traversal of its nodes' values.

7. Binary Tree Zigzag Level Order Traversal :

Given the root of a binary tree, return the zigzag level order traversal of its nodes' values. (i.e., from left to right, then right to left for the next level and alternate between).

8. Construct Binary Tree from Inorder and Postorder Traversal:

Given two integer arrays inorder and postorder where inorder is the inorder traversal of a binary tree and postorder is the postorder traversal of the same tree, construct and return the binary tree.

9. Kth Smallest Element in a BST :

Given the root of a binary search tree, and an integer k, return the k^{th} smallest value (1-indexed) of all the values of the nodes in the tree.

10. Populating Next Right Pointers in Each Node :

You are given a perfect binary tree where all leaves are on the same level, and every parent has two children. The binary tree has the following definition:

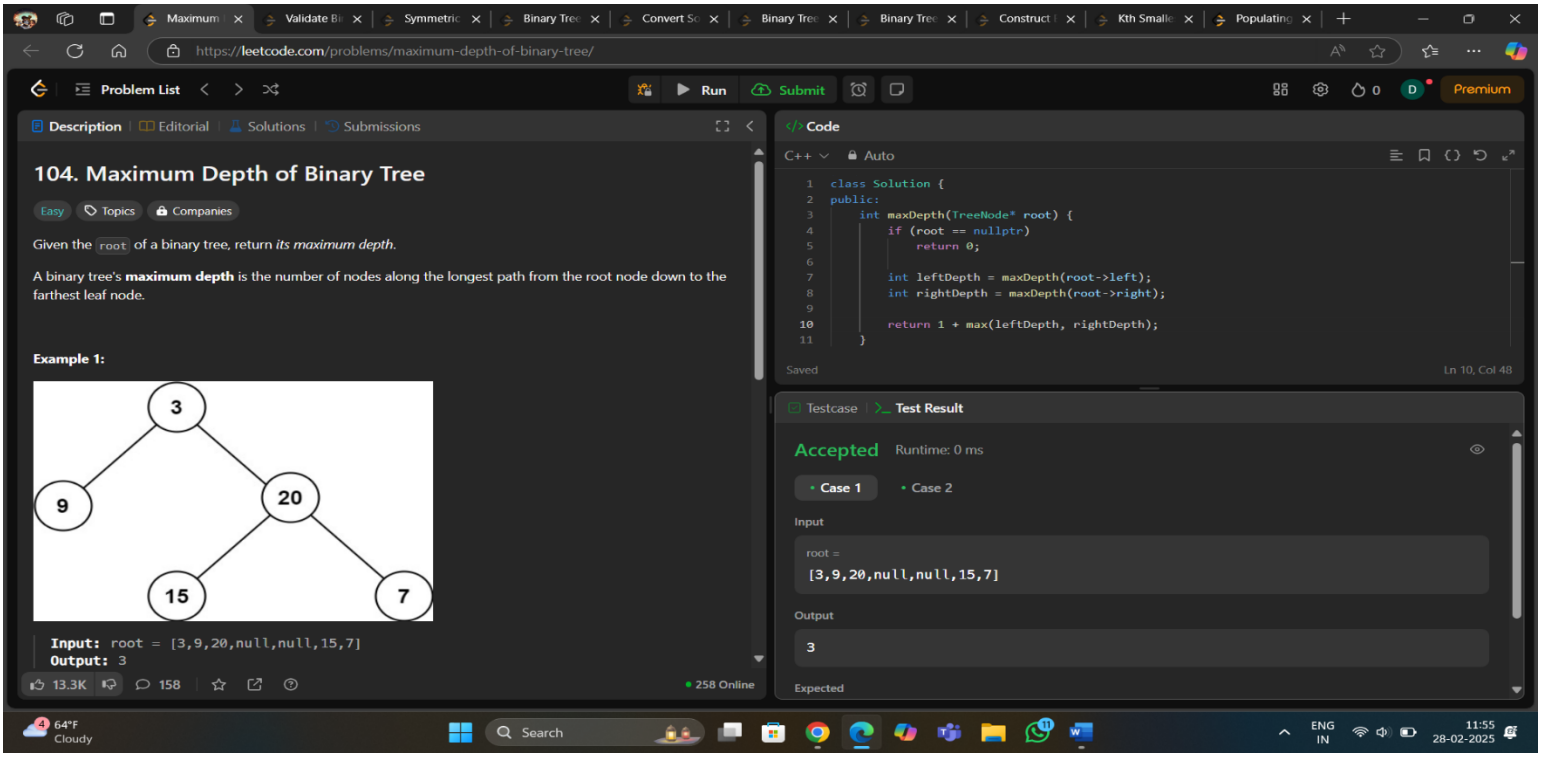
```
struct Node {
int val;
Node *left;
Node *right;
Node *next;
}
```

Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to NULL.
Initially, all next pointers are set to NULL.

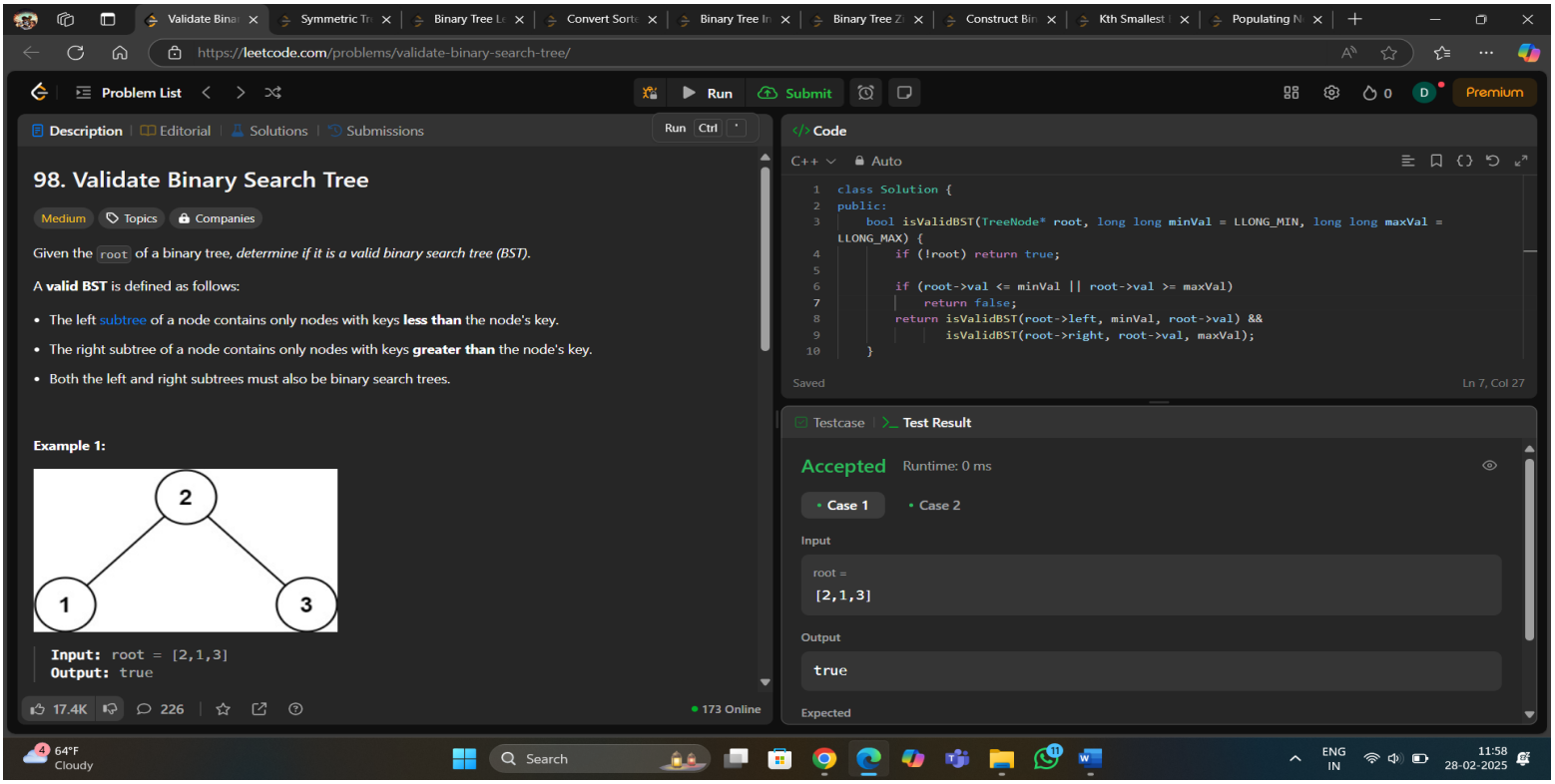
11.Objective :

1. Learn how to recursively or iteratively compute the height (or depth) of a binary tree.
2. Understand the properties of a Binary Search Tree (BST).
3. Understand mirror properties of a binary tree.
4. Learn Breadth-First Search (BFS) using queues.
5. Learn how to construct a height-balanced BST from a sorted array.
6. Understand inorder traversal (left-root-right).
7. Extend BFS level-order traversal to include zigzag ordering.
8. Understand tree reconstruction from traversal orders.
9. Use inorder traversal (sorted order) to efficiently find the kth smallest element.
10. Understand tree linking using level pointers.

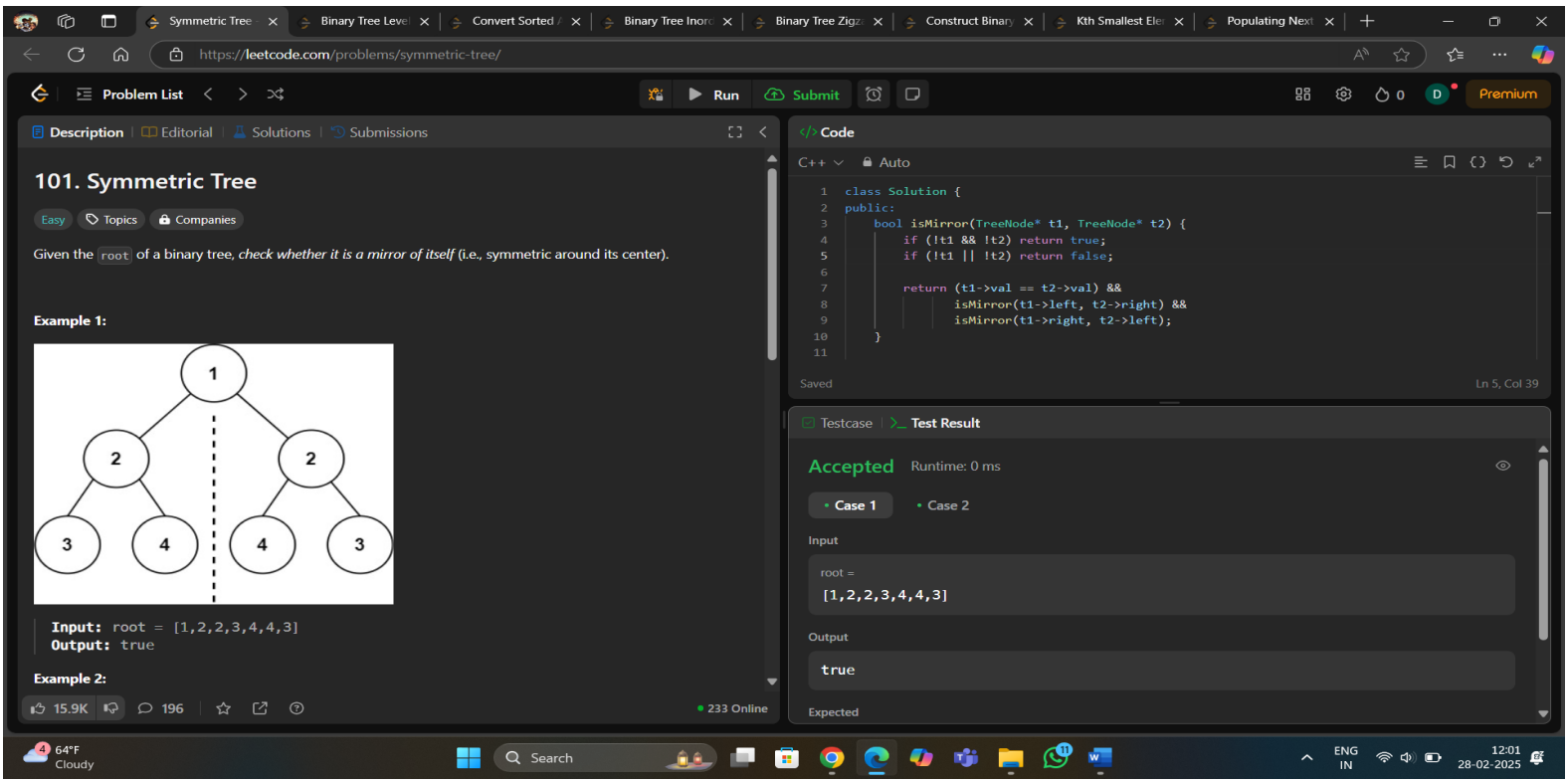
12.Implementation of Code/Output 1 :



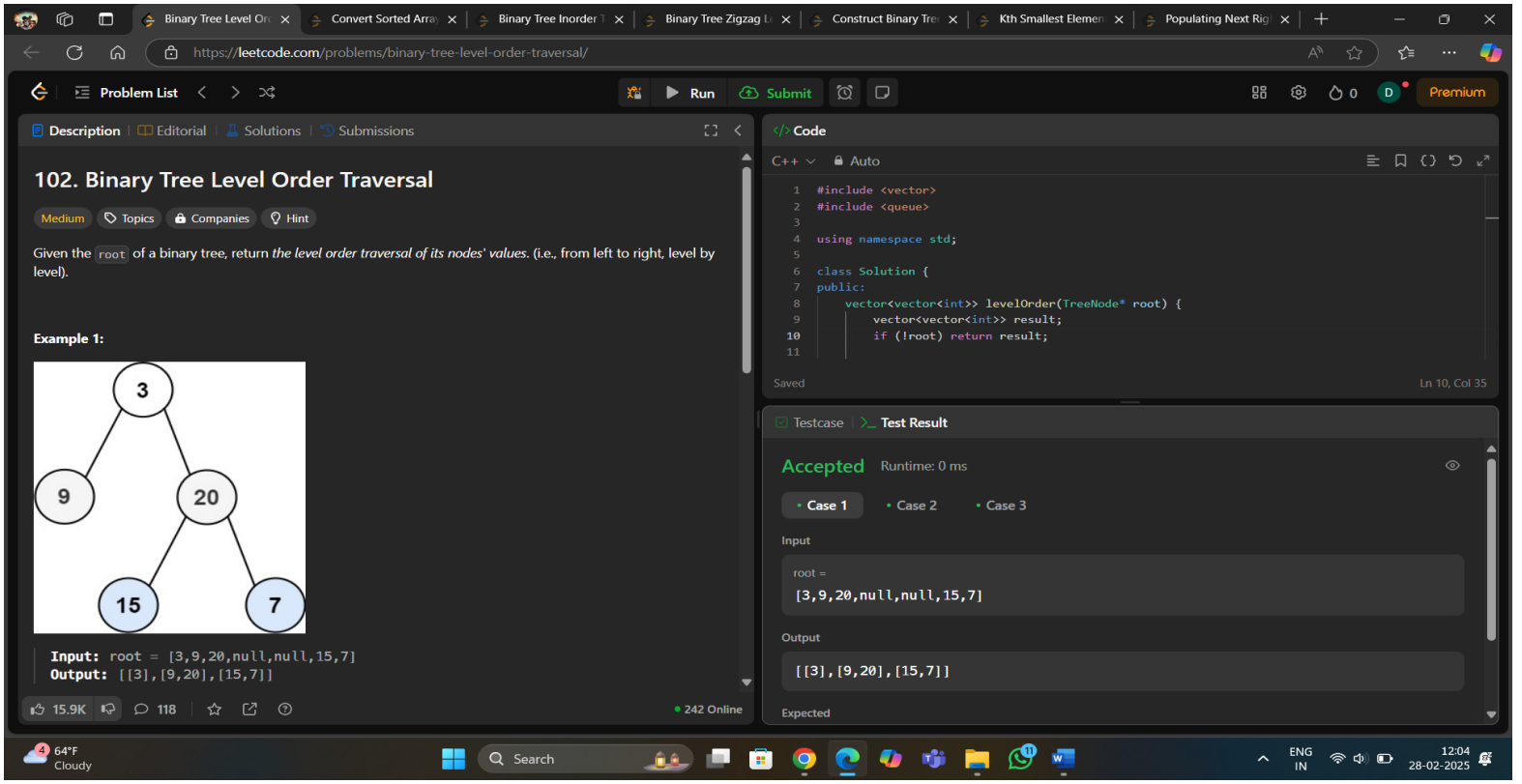
13.Code 2 :



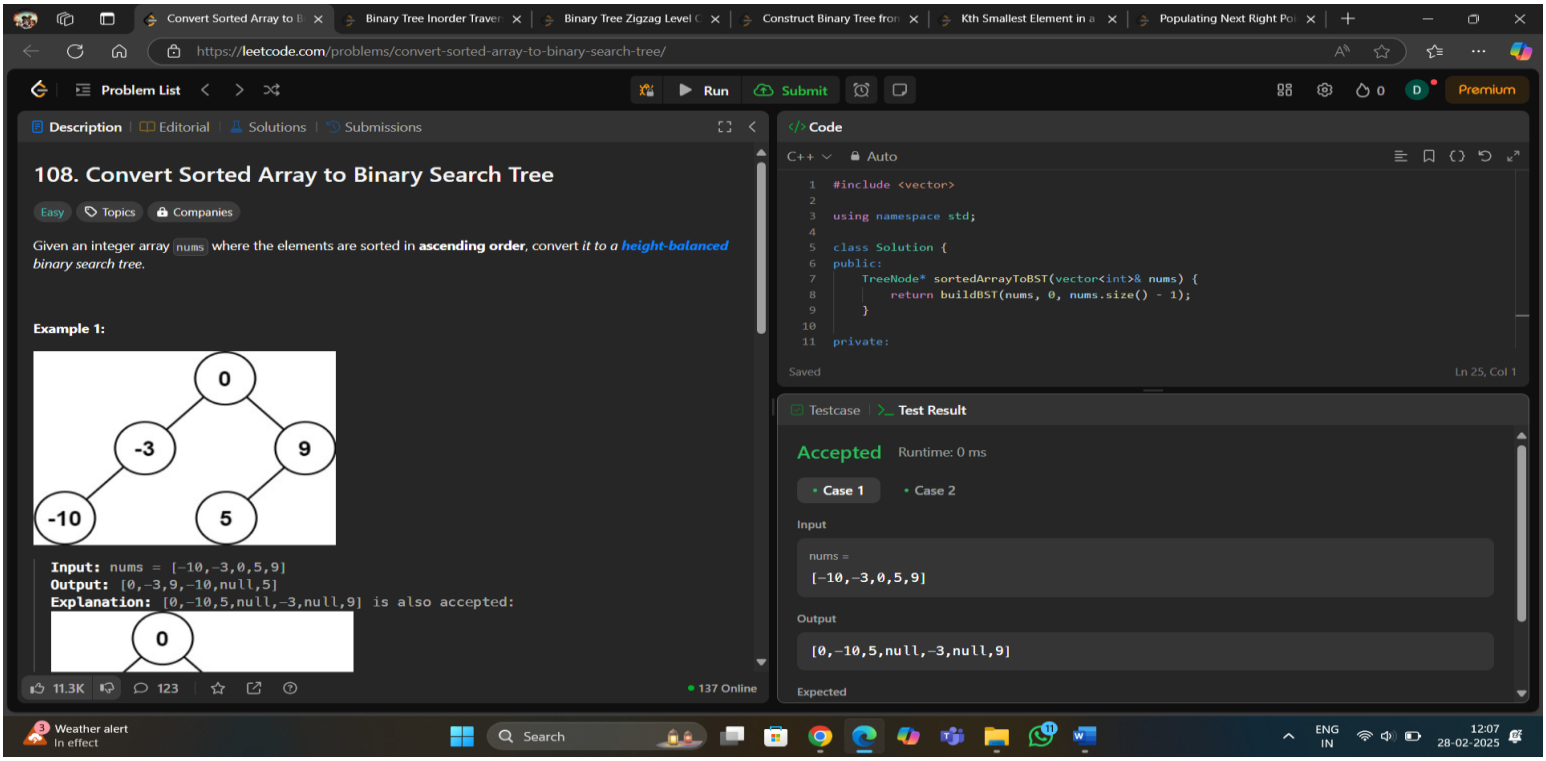
14.Code 3 :



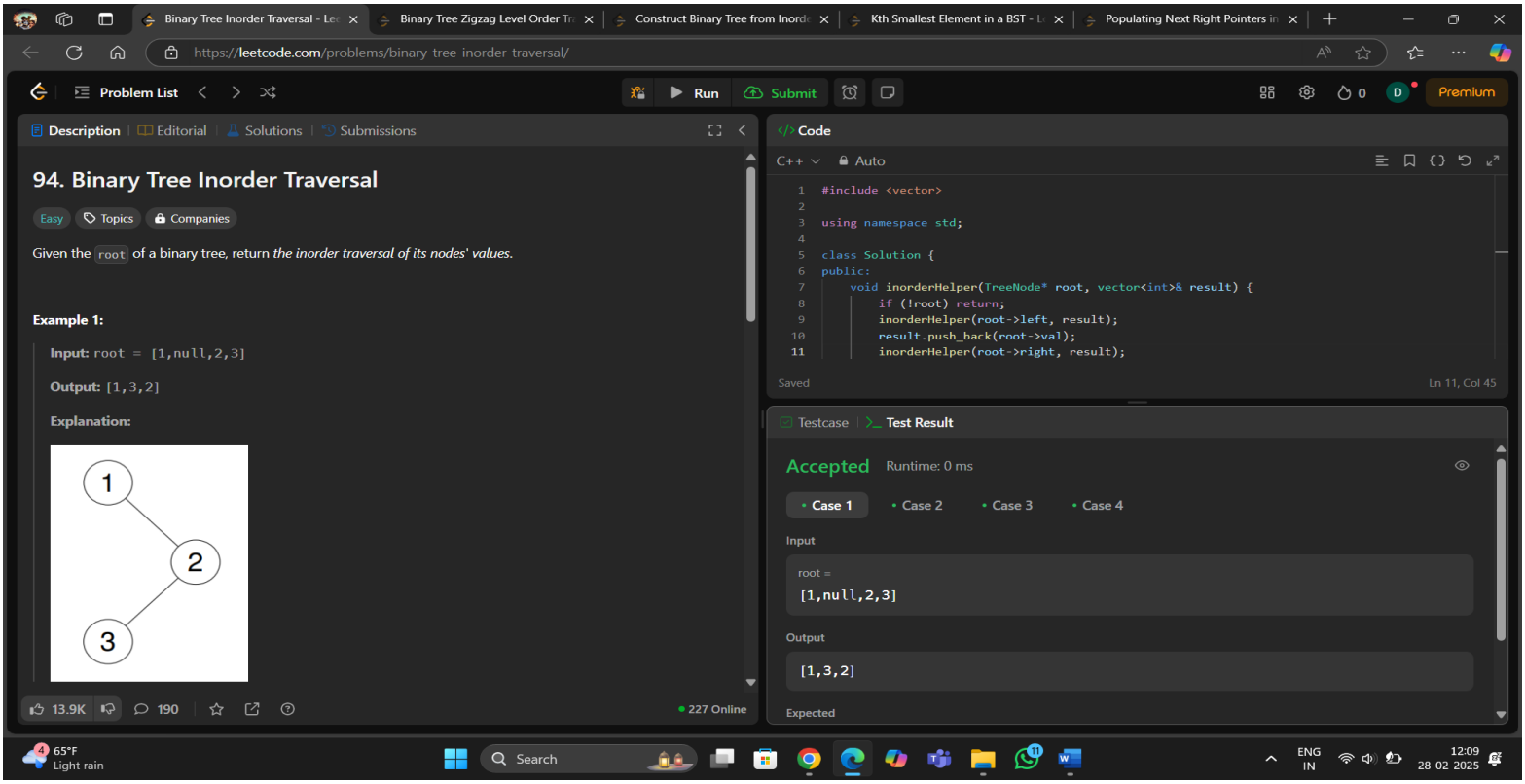
15.Code 4 :



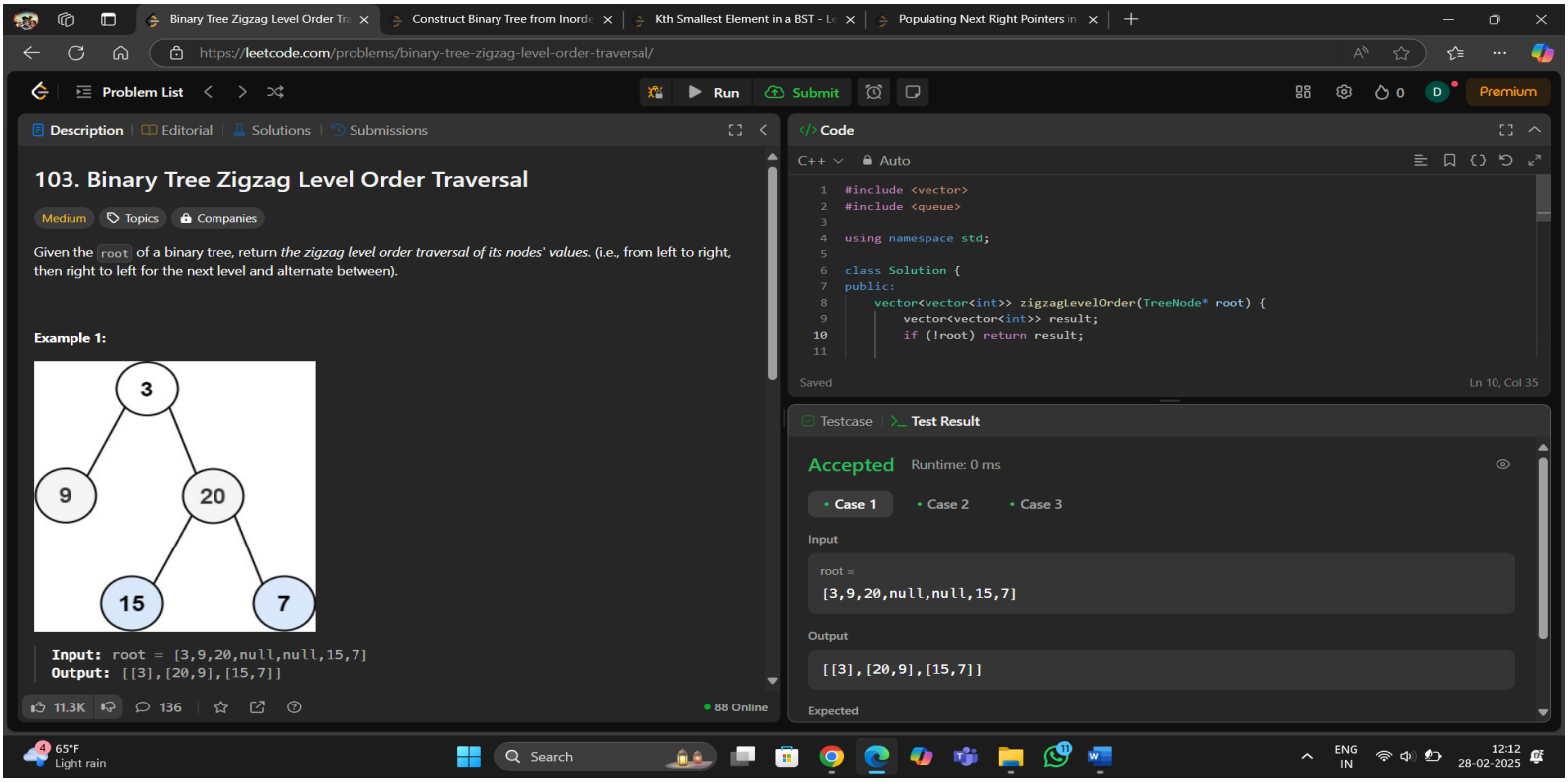
16.Code 5 :



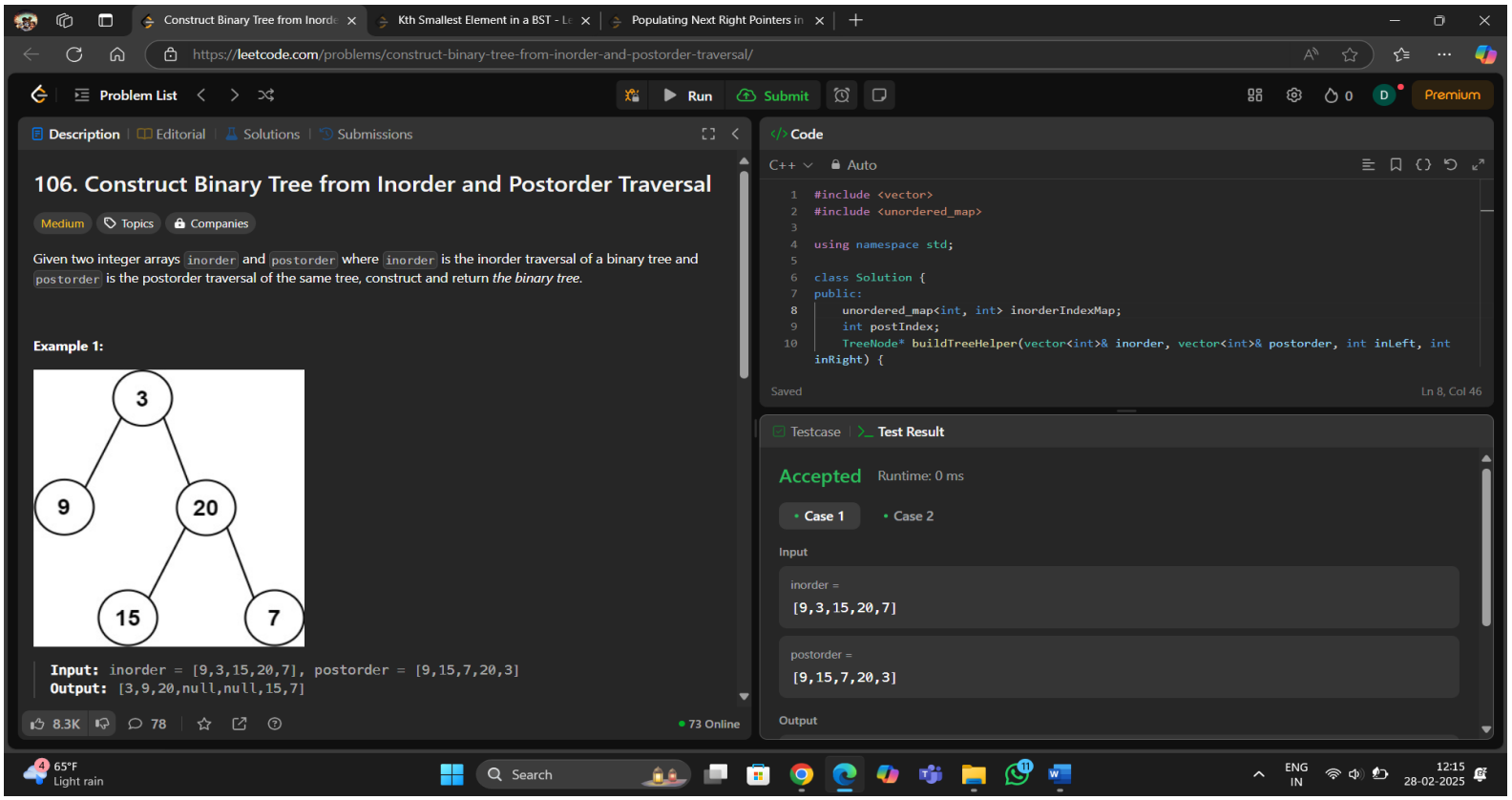
17.Code 6 :



18.Code 7 :



19.Code 8 :



20.Code 9:

Kth Smallest Element in a BST - Le | Populating Next Right Pointers in

https://leetcode.com/problems/kth-smallest-element-in-a-bst/

Problem List | Run | Submit | Premium

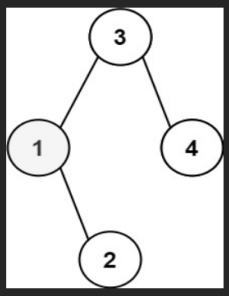
Description | Editorial | Solutions | Submissions

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* k^{th} *smallest value* ***(1-indexed)*** *of all the values of the nodes in the tree.*

Example 1:




```
Input: root = [3,1,4,null,2], k = 1
Output: 1
```

11.9K | 121 | 123 Online

Code

C++ | Auto

```
#include <stack>

class Solution {
public:
    int kthSmallest(TreeNode* root, int k) {
        stack<TreeNode*> s;
        TreeNode* current = root;

        while (current || !s.empty()) {
            while (current) {
                s.push(current);
```

Saved | Ln 27, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

root =
[3,1,4,null,2]

k =
1

Output

65°F Light rain | Search | ENG IN | 12:18 28-02-2025

21.Code 10 :

Populating Next Right Pointers in

https://leetcode.com/problems/populating-next-right-pointers-in-each-node/

Problem List | Run | Submit | Premium

Description | Editorial | Solutions | Submissions

116. Populating Next Right Pointers in Each Node

Medium | Topics | Companies

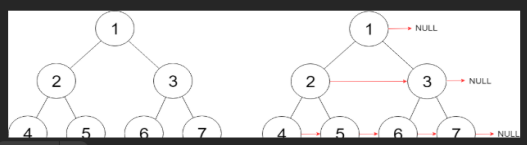
You are given a **perfect binary tree** where all leaves are on the same level, and every parent has two children. The binary tree has the following definition:

```
struct Node {
  int val;
  Node *left;
  Node *right;
  Node *next;
}
```

Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to `NULL`.

Initially, all next pointers are set to `NULL`.

Example 1:



10K | 72 | 72 Online

Code

C++ | Auto

```
#include <queue>

class Solution {
public:
    Node* connect(Node* root) {
        if (!root) return nullptr;

        queue<Node*> q;
        q.push(root);

        while (!q.empty()) {
```

Saved | Ln 6, Col 36

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

root =
[1,2,3,4,5,6,7]

Output

[1,#,2,3,#,4,5,6,7,#]

Expected

65°F Light rain | Search | ENG IN | 12:20 28-02-2025

22.Learning Outcome :

1. Concepts: Depth-first search (DFS), Recursion
2. A valid BST requires every node's left child to be smaller and right child to be larger.
3. A tree is symmetric if its left subtree is a mirror of the right subtree.
4. Concepts: Recursion, Divide & Conquer, Balanced BST
5. Concepts: Inorder traversal, Recursion, Iterative (Stack)
6. Concepts: Tree Reconstruction, Recursion, Hash Map Optimization