786

Week 6

1. **Height of Binary Tree**
   Write a function that returns the number of levels in a binary tree.
   An empty tree has height 0, and a tree with only one node has height 1.

2. **Determine if Two Trees Are Identical**
   Write a function that checks whether two binary trees are identical in both structure and node values.

3. **Mirror Tree**
   Modify a binary tree so that it becomes its mirror image by swapping the left and right child of every node.

4. **Symmetric Tree**
   Write a function that checks whether a binary tree is symmetric around its center.
   A tree is symmetric if the left subtree is a mirror image of the right subtree.

5. **Check for Balanced Tree**
   Write a function that checks whether the height difference between the left and right subtrees of every node is at most one.

6. **Children Sum Parent**
   Write a function that checks whether every non-leaf node's value is equal to the sum of the values of its left and right children.

7. **Array to BST**
   Given a sorted array, construct a height-balanced binary search tree such that the inorder traversal of the tree produces the same array.

8. **Nodes at Given Distance in a Binary Tree**
   Write a function that prints all nodes that are exactly k edges away from the root node.

9. **Inorder Successor in BST**
   Given a node in a binary search tree, find the node that appears immediately after it in an inorder traversal.

10. **Kth Largest Element in a BST**
    Write a function that returns the kth largest value stored in a binary search tree.

11. **Diameter of Tree**
    Write a function that returns the number of nodes on the longest path between any two nodes in a binary tree.

12. **Check if Subtree**
    Given two binary trees, determine whether the second tree exists as a subtree within the first tree.

13. **Check for BST**
    Write a function that checks whether a binary tree satisfies all the properties of a binary search tree.

14. **Single Valued Subtree**
    Count the number of subtrees in which all nodes contain the same value.

15. **Zigzag Tree Traversal**
    Traverse the binary tree level by level, alternating the direction of traversal at each level.

16. **Boundary Traversal**
    Print all boundary nodes of a binary tree in anti-clockwise order, starting from the root.

17. **Construct Binary Tree from Preorder and Inorder Traversal**
    Construct a binary tree using given preorder and inorder traversal sequences.

18. **Construct BST from Preorder Traversal**
    Construct a binary search tree using only its preorder traversal.

19. **Lowest Common Ancestor of a Binary Tree**
    Find the lowest common ancestor of two given nodes in a binary tree, where both nodes are guaranteed to exist.

20. **Ancestors in Binary Tree**
    Print all ancestor nodes of a given target node, starting from the root.