50 Tree Practice Problems

1. Create a Binary Tree from Level Order Input

Input:
Level Order: 1 2 3 -1 -1 4 5

Output:
Tree Structure as shown

2. Inorder Traversal of Binary Tree

Input:
Tree: [1, 2, 3, -1, -1, 4, 5]

Output:
Inorder: 2 1 4 3 5

3. Preorder Traversal of Binary Tree

Input:
Tree: [1, 2, 3, -1, -1, 4, 5]

Output:
Preorder: 1 2 3 4 5

4. Postorder Traversal of Binary Tree

Input:
Tree: [1, 2, 3, -1, -1, 4, 5]

Output:
Postorder: 2 4 5 3 1

5. Level Order Traversal

Input:
Tree: [1, 2, 3, -1, -1, 4, 5]

Output:
Level Order: 1 2 3 4 5

6. Count Total Nodes in Binary Tree

Input:
Tree: [1, 2, 3, 4, 5]

Output:
Total Nodes: 5

7. Calculate Height of Binary Tree

Input:
Tree: [1, 2, 3, -1, -1, 4, 5]

Output:
Height: 3

8. Count Leaf Nodes in Binary Tree

Input:
Tree: [1, 2, 3, -1, -1, -1, 4]

Output:
Leaf Nodes: 2

9. Check if Two Trees are Identical

Input:
Tree A: [1, 2, 3], Tree B: [1, 2, 3]

Output:
Identical: true

10. Mirror a Binary Tree

Input:
Original Tree: [1, 2, 3]

Output:
Mirrored Tree: [1, 3, 2]

11. Sum of All Nodes in Binary Tree

Input:
Tree: [1, 2, 3, 4]

Output:
Sum: 10

12. Find Maximum Element in Binary Tree

Input:
Tree: [1, 5, 3, 9, 2]

Output:
Maximum: 9

13. Find Minimum Element in Binary Tree

Input:
Tree: [8, 3, 10, 1, 6]

Output:
Minimum: 1

14. Print All Nodes at a Given Level

Input:
Tree: [1, 2, 3, 4, -1], Level: 2

Output:
Nodes at Level 2: 2 3

15. Check if a Binary Tree is Balanced

Input:
Tree: [1, 2, 3, 4, -1]

Output:
Balanced: true

16. Print Left View of Binary Tree

Input:
Tree: [1, 2, 3, 4, -1, -1, 5]

Output:
Left View: 1 2 4

17. Print Right View of Binary Tree

Input:
Tree: [1, 2, 3, 4, -1, -1, 5]

Output:
Right View: 1 3 5

18. Print Top View of Binary Tree

Input:
Tree: [1, 2, 3, 4, 5, 6, 7]

Output:
Top View: 4 2 1 3 7

19. Print Bottom View of Binary Tree

Input:
Tree: [1, 2, 3, 4, 5, 6, 7]

Output:
Bottom View: 4 2 6 3 7

20. Zig-Zag Level Order Traversal

Input:
Tree: [1, 2, 3, 4, 5, 6, 7]

Output:
Zig-Zag: [1], [3, 2], [4, 5, 6, 7]

21. Print Ancestors of a Given Node

Input:
Tree: [1, 2, 3, 4, 5], Target: 5

Output:
Ancestors of 5: 2 1

22. Lowest Common Ancestor (LCA) of Two Nodes

Input:
Tree: [3, 5, 1, 6, 2, 0, 8], Nodes: 6, 2

Output:
LCA: 5

23. Convert Binary Tree to Doubly Linked List

Input:
Tree: [10, 12, 15, 25, 30, 36]

Output:
DLL: 25 <-> 12 <-> 30 <-> 10 <-> 36 <-> 15

24. Print Nodes at Distance K from Root

Input:
Tree: [1, 2, 3, 4, 5], K = 2

Output:
Nodes at distance 2: 4 5

25. Path from Root to Given Node

Input:
Tree: [1, 2, 3, -1, 5], Node: 5

Output:
Path: 1 → 2 → 5

26. Maximum Path Sum in Binary Tree

Input:
Tree: [-10, 9, 20, -1, -1, 15, 7]

Output:
Max Path Sum: 42

27. Sum of Left Leaves

Input:
Tree: [3, 9, 20, -1, -1, 15, 7]

Output:
Left Leaf Sum: 24

28. Diameter of Binary Tree

Input:
Tree: [1, 2, 3, 4, 5]

Output:
Diameter: 4

29. Find All Root-to-Leaf Paths

Input:
Tree: [1, 2, 3, 4, 5]

Output:
Paths: 1→2→4, 1→2→5, 1→3

30. Check if a Binary Tree is a Subtree of Another

Input:
Tree A: [3, 4, 5, 1, 2], Tree B: [4, 1, 2]

Output:
Is Subtree: true

31. Construct Tree from Inorder and Preorder

Input:
Inorder: [4, 2, 5, 1, 3], Preorder: [1, 2, 4, 5, 3]

Output:
Constructed Binary Tree

32. Construct Tree from Inorder and Postorder

Input:
Inorder: [4, 2, 5, 1, 3], Postorder: [4, 5, 2, 3, 1]

Output:
Constructed Binary Tree

33. Binary Tree to Sum Tree

Input:
Tree: [10, -2, 6, 8, -4, 7, 5]

Output:
Sum Tree: [20, 4, 12, 0, 0, 0, 0]

34. Prune Tree Based on Path Sum Limit

Input:
Tree: [1, 2, 3], Limit = 4

Output:
Pruned Tree: [1, -1, 3]

35. Convert to Children Sum Property Tree

Input:
Tree: [50, 7, 2, 3, 5, 1, 30]

Output:
Updated Tree: [50, 8, 31, 3, 5, 1, 30]

36. Serialize and Deserialize Binary Tree

Input:
Tree: [1, 2, 3, -1, -1, 4, 5]

Output:
Serialized: 1,2,#,#,3,4,#,#,5,#,#

37. Check if Tree is Symmetric (Mirror of itself)

Input:
Tree: [1, 2, 2, 3, 4, 4, 3]

Output:
Symmetric: true

38. Count Nodes in a Complete Binary Tree (Efficient)

Input:
Tree: [1, 2, 3, 4, 5, 6]

Output:
Total Nodes: 6

39. Find Duplicate Subtrees

Input:
Tree: [1, 2, 3, 4, -1, 2, 4, -1, -1, 4]

Output:
Duplicate Subtrees: [4], [2,4]

40. Boundary Traversal of Binary Tree

Input:
Tree: [20, 8, 22, 4, 12, -1, 25, -1, -1, 10, 14]

Output:
Boundary: 20 8 4 10 14 25 22

41. Vertical Order Traversal

Input:
Tree: [1, 2, 3, 4, 5, 6, 7]

Output:
Vertical Order: [4], [2], [1,5,6], [3], [7]

42. Diagonal Traversal of Binary Tree

Input:
Tree: [8, 3, 10, 1, 6, -1, 14, -1, -1, 4, 7, 13]

Output:
Diagonals: [8,10,14], [3,6,7,13], [1,4]

43. Check if Tree is Foldable

Input:
Tree: [10, 7, 15, -1, 9, 11, -1]

Output:
Foldable: true

44. Build Binary Tree from String with Brackets

Input:
String: "4(2(3)(1))(6(5))"

Output:
Tree: Built Successfully

45. Sum of Nodes on Longest Path from Root to Leaf

Input:
Tree: [1, 2, 3, 4, 5, -1, 6]

Output:
Longest Path Sum: 10

46. Count Paths with Given Sum

Input:
Tree: [10, 5, -3, 3, 2, -1, 11, 3, -2, -1, 1], Sum = 8

Output:
Paths: 3

47. Binary Tree Cameras (Min Cameras to Monitor All Nodes)

Input:
Tree: [0, 0, -1, 0, 0]

Output:
Min Cameras: 1

48. Burning Tree from a Leaf Node (Time to Burn Whole Tree)

Input:
Tree: [1, 2, 3, -1, 4], Target Leaf = 4

Output:
Time: 3

49. Binary Tree Maximum Width

Input:
Tree: [1, 3, 2, 5, 3, -1, 9]

Output:
Max Width: 4

50. Convert Binary Tree to BST (Preserve Structure)

Input:
Tree: [10, 30, 15, 20]

Output:
BST: [15, 20, 30, 10]