**Coding challenge 1:**

**Invert Binary Tree**

\*\*Description (140 characters):\*\* Invert a binary tree by swapping left and right subtrees recursively.

**Problem Statement:**

Given the root of a binary tree, invert the tree by swapping every left child with its right child recursively.  
Return the root of the modified (inverted) tree.

**Input Format:** The input is a binary tree represented as a list (level order traversal).

**Constraints:**

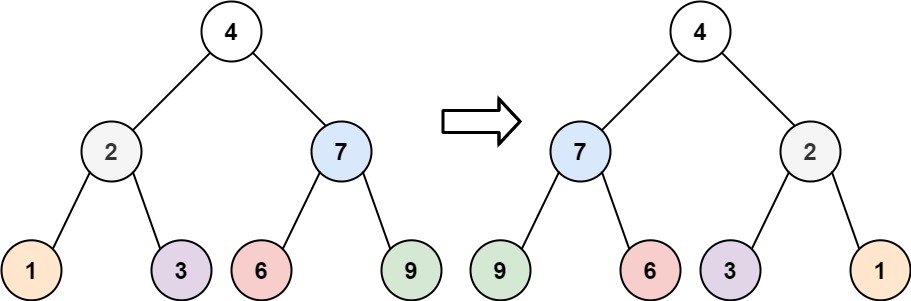
The number of nodes is in the range [0, 100].  
-100 <= Node.val <= 100

**Output Format:** Return the root of the inverted binary tree, represented as a list in level order.

**Tags:** Binary Tree, DFS, Recursion

**Sample Test Cases:**

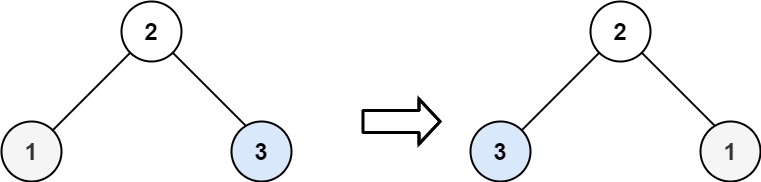
**Test case 1:**



Input: root = [4,2,7,1,3,6,9]

Output: [4,7,2,9,6,3,1]

**Test case 2:**



Input: root = [2,1,3]

Output: [2,3,1]

**Test case 3:**

Input: root = []

Output: []

**Test case 4:**

*Input:  
[1,2]*

*Output:  
[1,null,2]*

**Test case 5:**

*Input:  
[5,3,6,2,null,null,7]*

Output:  
[5,6,3,7,null,null,2]

**Coding Challenge 2:**

**Diameter of Binary Tree**

\*\*Description (140 characters):\*\* Find the longest path between any two nodes in a binary tree.

**Problem Statement:**

Given the root of a binary tree, return the length of the diameter of the tree. The diameter is the number of edges on the longest path between any two nodes in the tree. The path may or may not pass through the root.

**Input Format:** The input is a binary tree represented as a list (level order traversal).

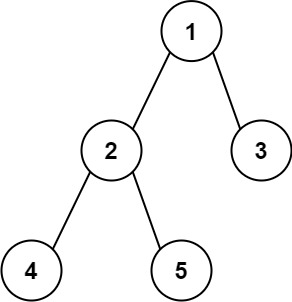
**Constraints:**

1 <= number of nodes <= 10^4  
-100 <= Node.val <= 100

**Output Format:** Return an integer denoting the diameter of the binary tree.

**Tags:** Binary Tree, DFS, Tree Traversal

**Example 1:**



**Input:** root = [1,2,3,4,5]

**Output:** 3

**Explanation:** 3 is the length of the path [4,2,1,3] or [5,2,1,3].

**Example 2:**

**Input:** root = [1,2]

**Output:** 1

Test case 3:

***Input: [1]***

***Output: 0***

Test case 4:

***Input:  
[1,2,3,4,null,null,5]***

***Output: 4***

Test case 5:

***Input:  
[1,2,3,null,null,4,5]***

***Output: 3***

**Coding Challenge 3:**

**Construct Binary Tree from Preorder and Inorder Traversal**

\*\*Description (140 characters):\*\* Construct a binary tree from its preorder and inorder traversals.

**Problem Statement:**

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

**Input Format:**

Two lists:  
- preorder: list of integers representing preorder traversal.  
- inorder: list of integers representing inorder traversal.

**Constraints:**

1 <= preorder.length <= 3000  
inorder.length == preorder.length  
-3000 <= preorder[i], inorder[i] <= 3000  
All values are unique.  
Each value in inorder also appears in preorder.

**Output Format:** Return the binary tree reconstructed from the given traversals in level order format.

**Tags:** Binary Tree, Recursion, Tree Construction, Divide and Conquer

**Example 1:**



**Input:** preorder = [3,9,20,15,7], inorder = [9,3,15,20,7]

**Output:** [3,9,20,null,null,15,7]

**Example 2:**

**Input:** preorder = [-1], inorder = [-1]

**Output:** [-1]

Test case 3:

***Input:  
preorder = [1,2], inorder = [2,1]***

***Output:  
[1,2]***

Test case 4:

***Input:  
preorder = [1,2,3], inorder = [1,2,3]***

***Output:  
[1,null,2,null,3]***

Test case 5:

***Input:  
preorder = [3,2,1], inorder = [1,2,3]***

***Output:  
[3,2,null,1]***