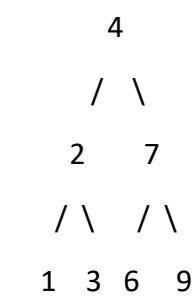
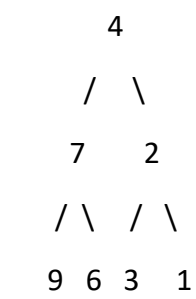
Invert Binary Tree LeetCode

Given the root of a binary tree, invert the tree, and return *its root*.

Example:



Output:



Approach 1: Recursively invert a binary tree

- The **invertTreeRecursively** function takes a binary tree's root as input and inverts the tree recursively.
- In the base case, it returns if the tree is empty (null).
- For each non-null node, it swaps the left and right subtrees by using the **swap** function from the C++ Standard Library.
- It then recursively inverts the left and right subtrees.
- Finally, it returns the root of the inverted tree.

Time Complexity: O(N), where N is the number of nodes in the binary tree. You visit each node once.

Space Complexity: O(H), where H is the height of the binary tree due to the function call stack.

Approach 2: Iteratively invert a binary tree using a stack

- The **invertTreeIteratively** function takes a binary tree's root as input and inverts the tree iteratively using a stack.
- In the base case, it returns if the tree is empty (null).
- It initializes a stack with the root node and enters a while loop.
- In each iteration, it pops a node from the stack, swaps its left and right subtrees, and pushes the left and right children onto the stack for further processing.
- The loop continues until the stack is empty.
- Finally, it returns the root of the inverted tree.

Time Complexity: O(N), where N is the number of nodes in the binary tree. You visit each node once.

Space Complexity: O(W), where W is the maximum width of the binary tree at any level due to the stack.