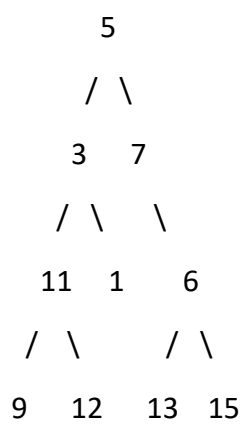
Find a Corresponding Node of a Binary Tree in Clone Tree LeetCode

Given two binary trees original and cloned and given a reference to a node target in the original tree.

The cloned tree is a **copy of** the original tree.

Return *a reference to the same node* in the cloned tree.

Example:



Target: 13

Output: true

Approach 1: Recursive approach to search for a target node in a cloned binary tree.

- In the **searchNodeHelper** function, you recursively search for a target node in a cloned binary tree.
- The base case checks if the current node is null. If it is, the function returns.
- If the current node's value matches the target's value, the answer (**ans**) is updated with the current node.
- The function then recursively searches the left and right subtrees.
- The **searchNodeRecursively** function initializes the answer as null and calls the helper function, returning the answer at the end.

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: Optimized recursive approach to search for a target node in a cloned binary tree.

- In the **searchNodeRecursivelyOptimized** function, you recursively search for a target node in a cloned binary tree.
- The base case checks if the current node is null. If it is, the function returns null.
- If the current node's value matches the target's value, the current node is returned as the answer.
- The function then recursively searches the left subtree, and if a result is found, it's returned.
- If no result is found in the left subtree, the function recursively searches the right subtree.
- This approach avoids unnecessary recursive calls when the target node is found.

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 3: Iterative approach to search for a target node in a cloned binary tree.

- The **searchNode** function uses an iterative approach to search for a target node in a cloned binary tree.
- It initializes a stack with the root node and enters a while loop.
- In each iteration, it pops a node from the stack and checks if its value matches the target's value.
- If a match is found, the node is returned as the answer.
- If no match is found, the left and right children are pushed onto the stack for further processing.
- The loop continues until the stack is empty or a match is found.

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.