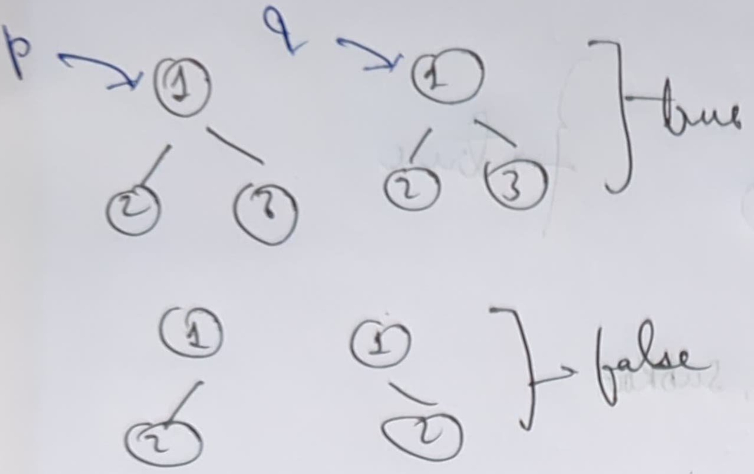
Identical $\overset{\text{Tree}}{}$(same tree) | Subtree of another tree



return type
↳ bool

identical → node Structure
value

(to check in terms of structure & in terms of values they are same or not)

Pseudo code

{ p → root of ~~to~~ 1st tree
q → " " 2nd tree

```
bool isIdentical(p, q){
    if (p==NULL || q==NULL)
        return p==q  → True agr dono NULL else False
    isLeftsame = isIdentical(p→left, q→left)
    isRightsame = isIdentical(p→right, q→right)
    return isleftsame && isRightsame && p→val == q→val
```

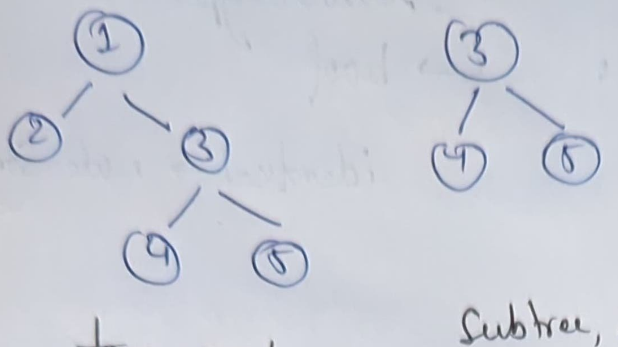
Base Case

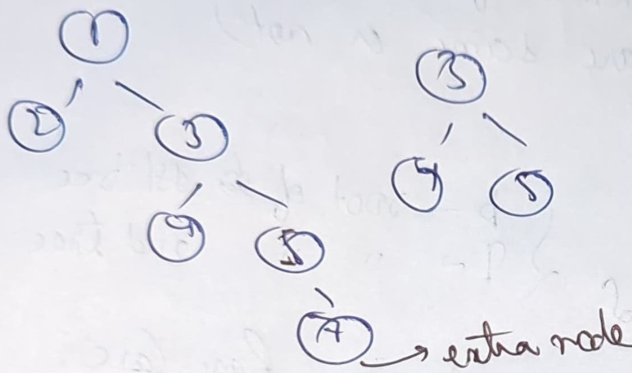
p	q	
NULL	value	→ false
value	NULL	→ false
NULL	NULL	→ true

TC: O(n)

SC: O(h)

Subtree Tree of Another Tree

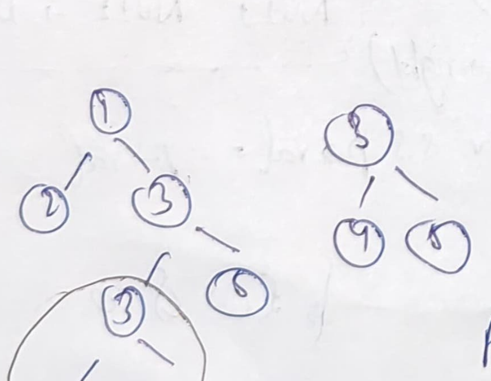




(should be exactly identical)

Approach :- ① find subRoot in main tree
② check isIdentical

$$\begin{tikzcd} & \text{isIdentical} \arrow[dl, "\text{for}"'] \arrow[dr] & \\ \text{subRoot} & & \text{main tree root} \end{tikzcd}$$



{LC : 572

Pseudo Code:

```
bool isSubtree (root, SubRoot){
    if (root == NULL || SubRoot == NULL)
        return root == SubRoot
    if (root → val == SubRoot → val &&
            isIdentical (root, SubRoot))
        return true;
    return isSubtree(root→left, SubRoot)
        || isSubtree(root→right, SubRoot)
```

572. Subtree of Another Tree

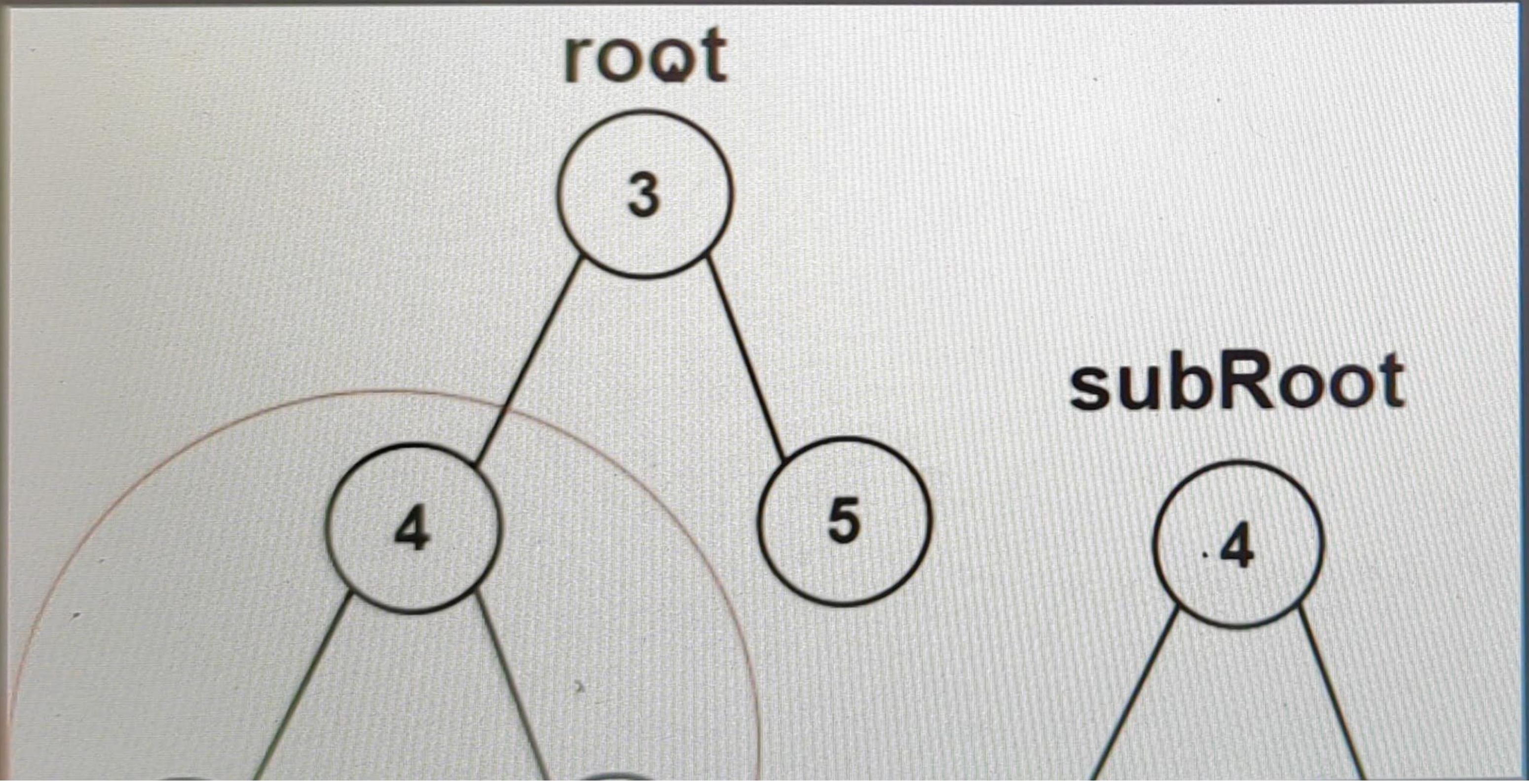
Solved ✓

Easy   Topics   Companies   Hint

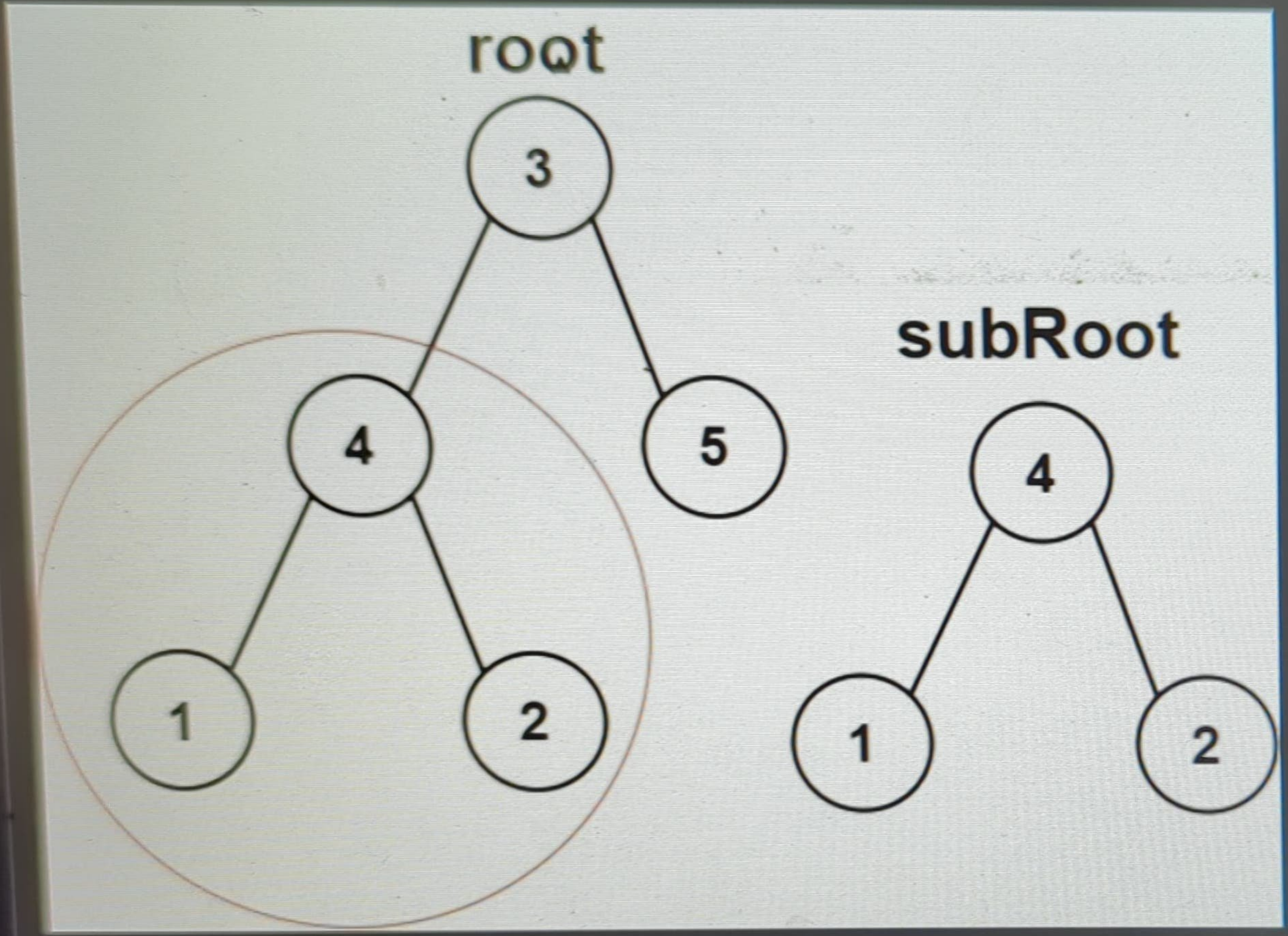
Given the roots of two binary trees `root` and `subRoot`, return `true` if there is a subtree of `root` with the same structure and node values of `subRoot` and `false` otherwise.

A subtree of a binary tree `tree` is a tree that consists of a node in `tree` and all of this node's descendants. The tree `tree` could also be considered as a subtree of itself.

Example 1:

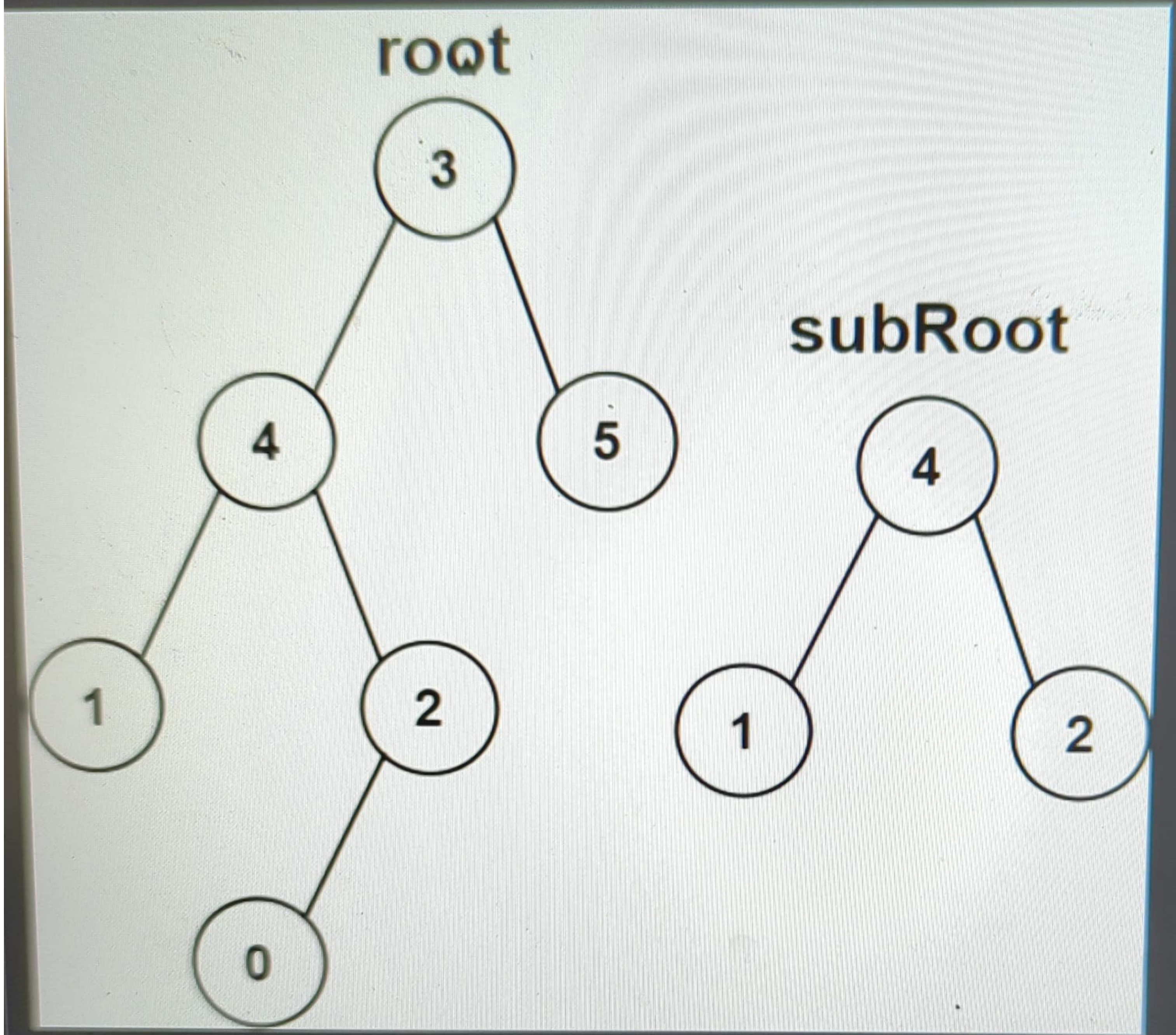


Example 1:

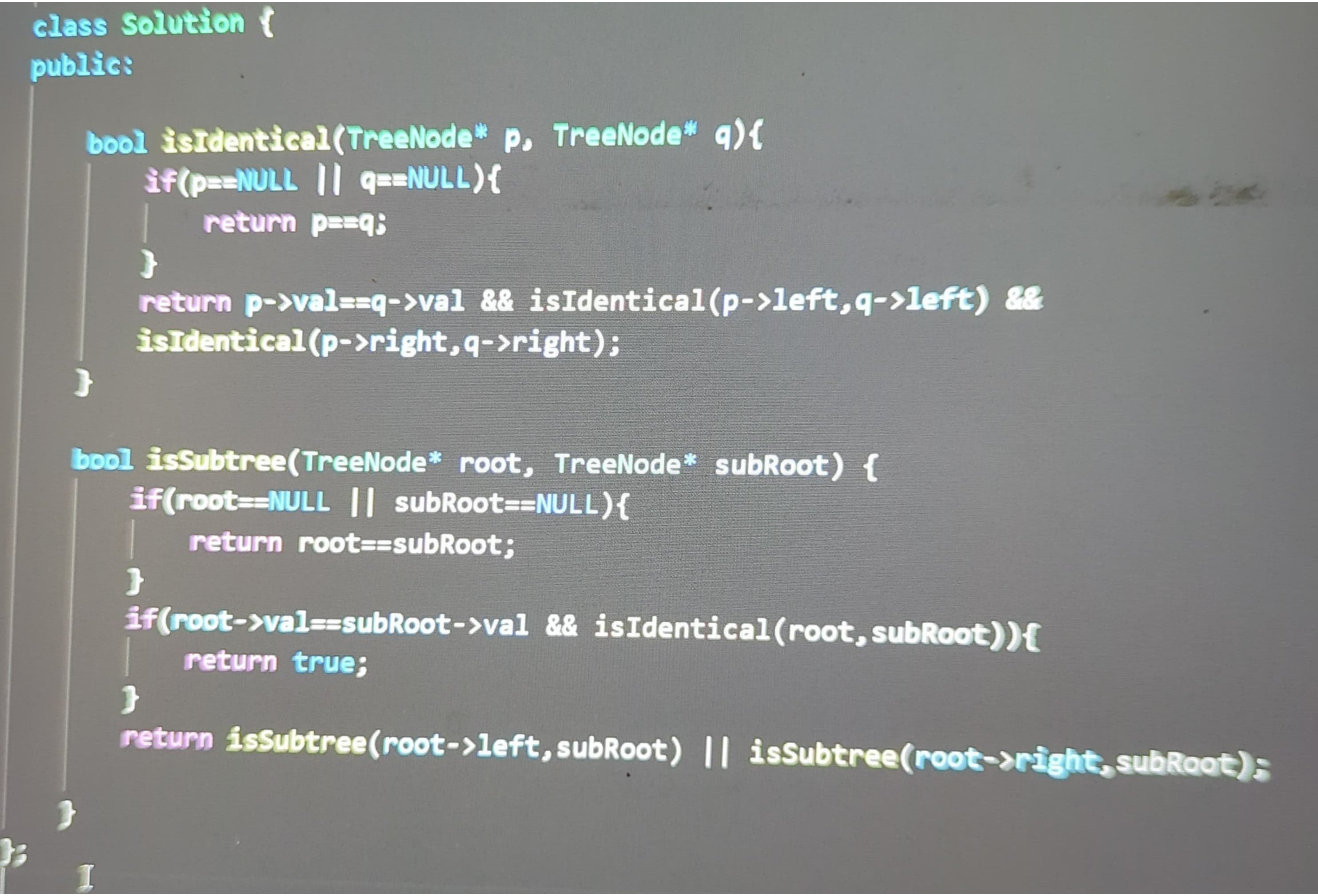


```
Input: root = [3,4,5,1,2], subRoot = [4,1,2]
Output: true
```

xample 2:



```
Input: root = [3,4,5,1,2,null,null,null,null,0], subRoot = [4,1,2]
Output: false
```



```cpp
class Solution {
public:

    bool isIdentical(TreeNode* p, TreeNode* q){
        if(p==NULL || q==NULL){
            return p==q;
        }
        return p->val==q->val && isIdentical(p->left,q->left) &&
        isIdentical(p->right,q->right);
    }

    bool isSubtree(TreeNode* root, TreeNode* subRoot) {
        if(root==NULL || subRoot==NULL){
            return root==subRoot;
        }
        if(root->val==subRoot->val && isIdentical(root,subRoot)){
            return true;
        }
        return isSubtree(root->left,subRoot) || isSubtree(root->right,subRoot);
    }
};
```