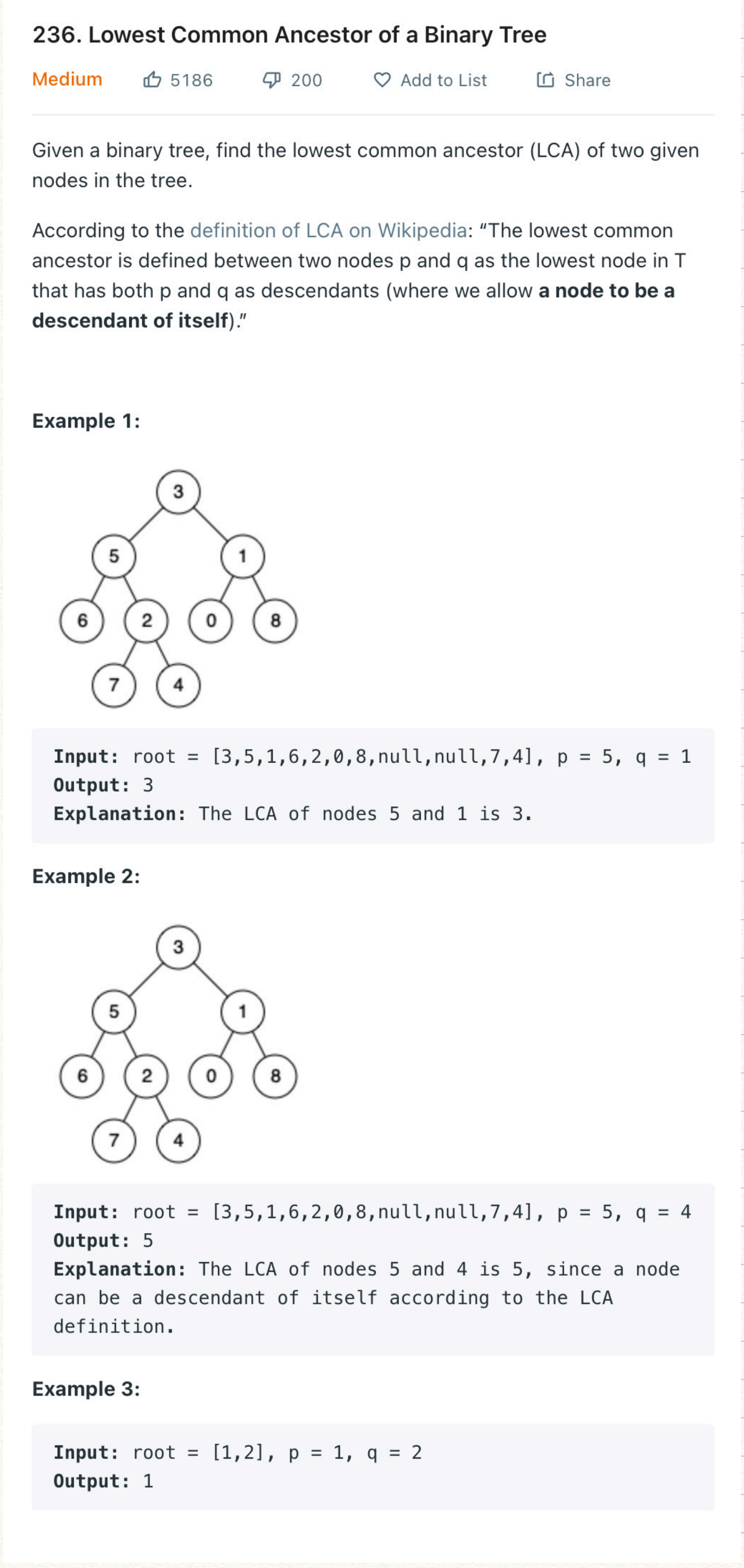


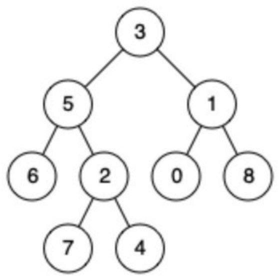
236. Lowest Common Ancestor of a Binary Tree

Medium 5186 200 Add to List Share

Given a binary tree, find the lowest common ancestor (LCA) of two given nodes in the tree.

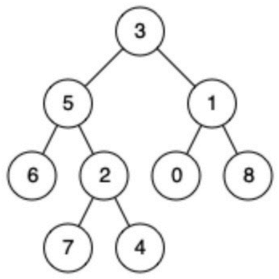
According to the definition of LCA on Wikipedia: "The lowest common ancestor is defined between two nodes p and q as the lowest node in T that has both p and q as descendants (where we allow **a node to be a descendant of itself**)."

Example 1:



```
Input: root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 1
Output: 3
Explanation: The LCA of nodes 5 and 1 is 3.
```

Example 2:



```
Input: root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 4
Output: 5
Explanation: The LCA of nodes 5 and 4 is 5, since a node
can be a descendant of itself according to the LCA
definition.
```

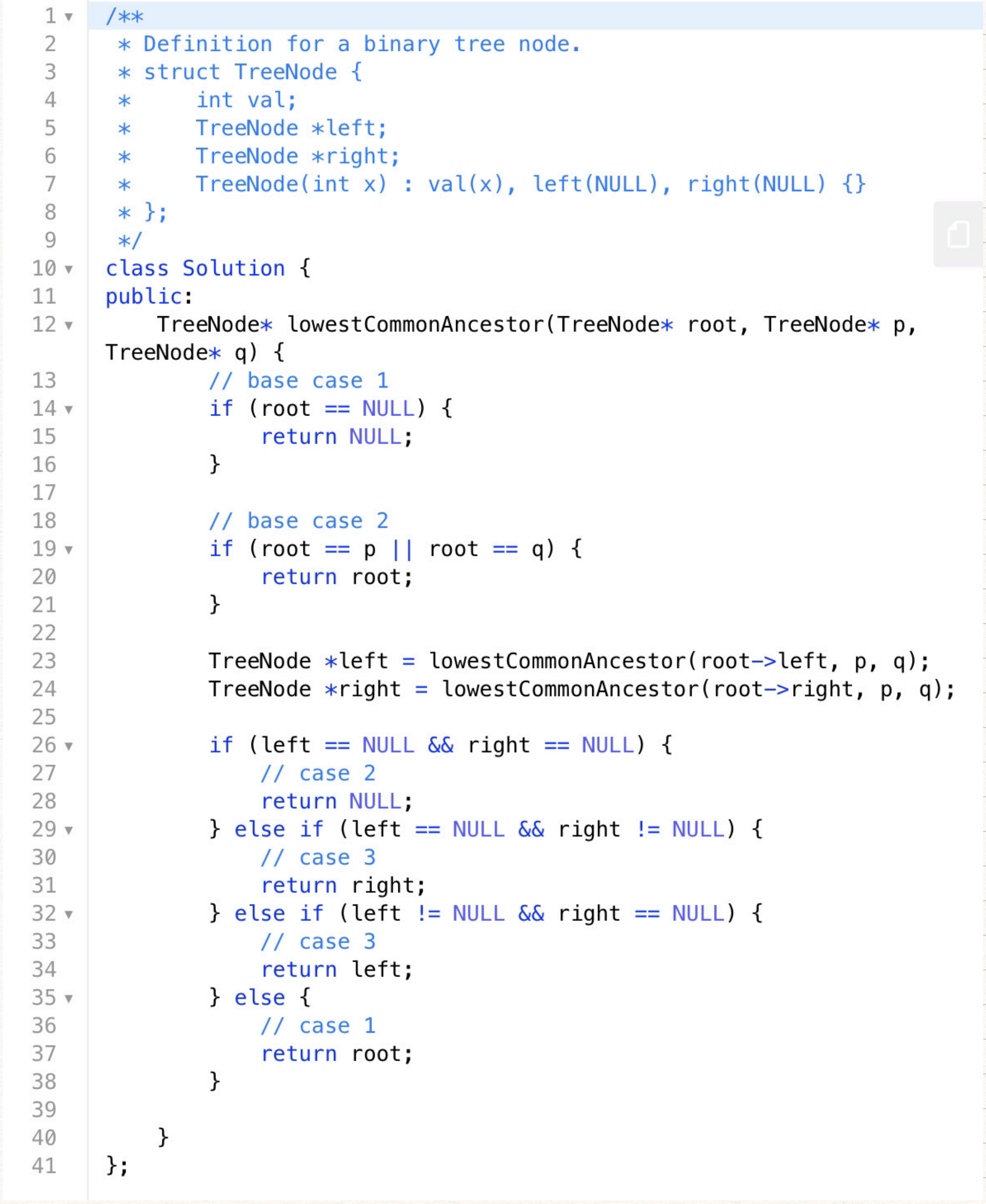
Example 3:

```
Input: root = [1,2], p = 1, q = 2
Output: 1
```

Case 1: If p and q are children of root, left and right are p and q seperately.

Case 2: If p and q are not children of root, return null.

Case 3: If one of p and q is child of root, return root.



```cpp
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
class Solution {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
        // base case 1
        if (root == NULL) {
            return NULL;
        }

        // base case 2
        if (root == p || root == q) {
            return root;
        }

        TreeNode *left = lowestCommonAncestor(root->left, p, q);
        TreeNode *right = lowestCommonAncestor(root->right, p, q);

        if (left == NULL && right == NULL) {
            // case 2
            return NULL;
        } else if (left == NULL && right != NULL) {
            // case 3
            return right;
        } else if (left != NULL && right == NULL) {
            // case 3
            return left;
        } else {
            // case 1
            return root;
        }

    }
};
```