Lab 6. Introduction to algorithms

This is sixth HW for CS 566.

Task 1. Solve the problem "Preorder Traversal" from https://leetcode.com/problems/binary-tree-preorder-traversal/description/ using Python3.

Use the box below, to paste the working code. The format of the code should be identical to LeetCode platform. (4 points)

```
from typing import Optional, List

class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def preorderTraversal(self, root: Optional[TreeNode]) -> List[int]:
        result = []
        def helper(root):
            if root:
                result.append(root.val)
                helper(root.left)
                helper(root.right)
        helper(root)
        return result
```

Do not modify the testing code below. If you get message "Mistake in test case #", it means that you algorithm is incorrect.

```
#test_case_1
root = TreeNode(1, None, TreeNode(2, TreeNode(3), None))
expected = [1, 2, 3]
actual = Solution().preorderTraversal(root)
assert actual==expected, "Mistake in test case 1"
print('OK')
```

```
OK
```

Write analysis of the Memory Complexity and Time Complexity using Aymptotic Notation O. (1 point)

Memory Analysis: O(h), where h is height of tree (can be log(n) or n)

Time Analysis: O(n)

Task 2. Solve the problem "Inorder Traversal" from https://leetcode.com/problems/binary-tree-inorder-traversal/description/ using Python3.

Use the box below, to paste the working code. The format of the code should be identical to LeetCode platform. (4 points)

```
from typing import Optional, List

class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def inorderTraversal(self, root: Optional[TreeNode]) -> List[int]:
        result = []
        def helper(root):
            if root:
                helper(root.left)
                result.append(root.val)
                helper(root.right)
        helper(root)
        return result
```

```
#test_case_1
root = TreeNode(1, None, TreeNode(2, TreeNode(3), None))
expected = [1, 3, 2]
actual = Solution().inorderTraversal(root)
assert actual==expected, "Mistake in test case 1"
print('OK')
```

```
OK
```

Write analysis of the Memory Complexity and Time Complexity using Asymptotic Notation O. (1 point)

Memory Analysis: O(h), where h is height of tree

Time Analysis: O(n)

Task 3. Solve the problem "PostOrder Traversal" from https://leetcode.com/problems/binary-tree-postorder-traversal/ using Python3.

Use the box below, to paste the working code. The format of the code should be identical to LeetCode platform. (4 points)

```
from typing import Optional, List

class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

class Solution:
    def postorderTraversal(self, root: Optional[TreeNode]) -> List[int]:
        result = []
        def helper(root):
            if root:
                helper(root.left)
                helper(root.right)
                result.append(root.val)
        helper(root)
        return result
```

```
#test_case_1
root = TreeNode(1, None, TreeNode(2, TreeNode(3), None))
expected = [3, 2, 1]
actual = Solution().postorderTraversal(root)
assert actual==expected, "Mistake in test case 1"
print('OK')
```

```
OK
```

Write analysis of the Memory Complexity and Time Complexity using Asymptotic Notation O. (1 point)

Memory Analysis: O(h), where h is height of tree

Time Analysis: O(n)