# 145. Binary Tree Postorder Traversal

Given the `root` of a binary tree, return *the postorder traversal of its nodes' values*.

```
class Pair:
    def __init__(self,node,state):
        self.node = node
        self.state = state
class Solution:
    def postorderTraversal(self, root: TreeNode) -> List[int]:
        if root is None:
            return
        pair = Pair(root,1)
        stack = [pair]
        res = []
        while len(stack)>0:
            temp= stack[-1]
            if temp.state==1:
                temp.state = temp.state +1
                if temp.node.left:
                    stack.append(Pair(temp.node.left,1))
            elif temp.state==2:
                temp.state = temp.state +1
                if temp.node.right:
                    stack.append(Pair(temp.node.right,1))
            else:
                res.append(temp.node.val)
                stack.pop()
        return res
```

```
def postorderTraversal(self, root: TreeNode) -> List[int]:
        if not root:
            return []
        # Stack of nodes to process. "True" only when children trees have
been traversed.
        stack = [(root, False)]
        result = []
        while stack:
            node, done = stack.pop()
            if done:
                result.append(node.val)
            else:
                # For post-order traversal, need to first visit left then
right before node is "done", so add them in reverse order to the stack.
                # By changing the order here we could achieve pre- or in-
order as well.
                stack.append((node, True))
                if node.right:
                    stack.append((node.right, False))
                if node.left:
                    stack.append((node.left, False))
        return result
```