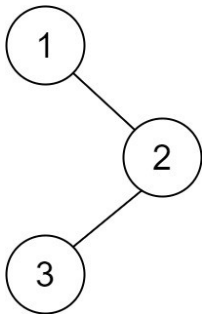
145. Binary Tree Postorder Traversal

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

Example 1:




```
Input: root = [1,null,2,3]
Output: [3,2,1]
```

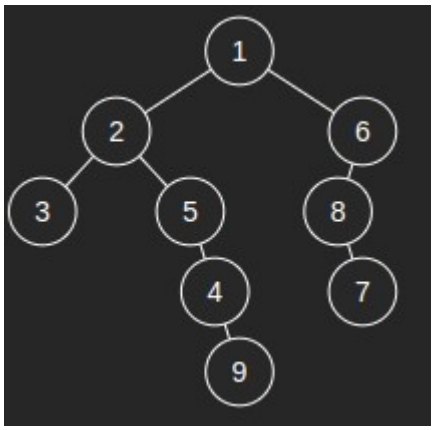
Example 2:

```
Input: root = []
Output: []
```

Example 3:

```
Input: root = [1]
Output: [1]
```

Example 4:




```
Input:
[1,2,6,3,5,8,null,null,null,null,4,null,7,null
,9]
Output: [3,9,4,5,2,7,8,6,1]
```

Constraints:

- The number of the nodes in the tree is in the range `[0, 100]`.
- `-100 <= Node.val <= 100`

145. Binary Tree Postorder Traversal

```
/*
    Recursion DFS
    Time complexity: O(n)
    Space complexity: O(n)
    n: #nodes in the binary tree
*/
typedef std::vector<int> vi;
class Solution {
    public:
        vi ans={};
    public:
        vi postorderTraversal(TreeNode* root) {
            if(!root) return ans;
            postorderTraversal(root->left);
            postorderTraversal(root->right);
            ans.push_back(root->val);
            return ans;
        }
};
```

145. Binary Tree Postorder Traversal

```
/*
    Iterative DFS
    Time complexity: O(n)
    Space complexity: O(n)
    n: #nodes in the binary tree
*/
typedef std::vector<int> vi;
class Solution {
    public:
        vi postorderTraversal(TreeNode* root) {
            if(!root) return {};
            vi ans;
            std::stack<TreeNode*> st;
            st.push(root);
            while(!st.empty()){
                auto cur_node=st.top();
                st.pop();
                ans.push_back(cur_node->val);
                if(cur_node->left) st.push(cur_node->left);
                if(cur_node->right) st.push(cur_node->right);
            }
            reverse(ans.begin(),ans.end());
            return ans;
        }
};
```

145. Binary Tree Postorder Traversal

```
/*
   Morris: adapted to post traversal
   Time complexity: O(n)
   Space complexity: O(1)
   n: #nodes in the binary tree
*/
typedef std::vector<int> vi;
class Solution {
    public:
        vi postorderTraversal(TreeNode* root) {
           vi ans;
            TreeNode* cur=root;
            while (cur){
                TreeNode* ptr = cur->right;
                if (!ptr) {
                    ans.push_back(cur->val);
                    cur=cur->left;
                }
                else {
                    while (ptr->left && ptr->left!=cur) ptr=ptr->left;

                    if (!ptr->left) {
                        ans.push_back(cur->val);
                        ptr->left = cur;
                        cur=cur->right;
                    }
                    else {
                        ptr->left = NULL;
                        cur=cur->left;
                    }
                }
            }

            reverse(ans.begin(), ans.end());
            return ans;
        }
};
```