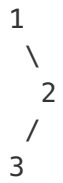


## Binary Tree Preorder Traversal

Given a binary tree, return the *preorder* traversal of its nodes' values.

For example:

Given binary tree `{1,#,2,3}`,



return `[1,2,3]`.

**Note:** Recursive solution is trivial, could you do it iteratively?

## Solution 1

Note that in this solution only right children are stored to stack.

```
public List<Integer> preorderTraversal(TreeNode node) {  
    List<Integer> list = new LinkedList<Integer>();  
    Stack<TreeNode> rights = new Stack<TreeNode>();  
    while(node != null) {  
        list.add(node.val);  
        if (node.right != null) {  
            rights.push(node.right);  
        }  
        node = node.left;  
        if (node == null && !rights.isEmpty()) {  
            node = rights.pop();  
        }  
    }  
    return list;  
}
```

written by [pavel-shlyk](#) original link [here](#)

## Solution 2

1. Create an empty stack, Push root node to the stack.
2. Do following while stack is not empty.
  - 2.1. pop an item from the stack and print it.
  - 2.2. push the right child of popped item to stack.
  - 2.3. push the left child of popped item to stack.

```
class Solution {
public:
    vector<int> preorderTraversal(TreeNode *root) {
        stack<TreeNode*> nodeStack;
        vector<int> result;
        //base case
        if(root==NULL)
            return result;
        nodeStack.push(root);
        while(!nodeStack.empty())
        {
            TreeNode* node= nodeStack.top();
            result.push_back(node->val);
            nodeStack.pop();
            if(node->right)
                nodeStack.push(node->right);
            if(node->left)
                nodeStack.push(node->left);
        }
        return result;
    }
};
```

written by [Deepalaxmi](#) original link [here](#)

## Solution 3

```
class Solution {
public:
    vector<int> preorderTraversal(TreeNode *root) {
        if (root==NULL) {
            return vector<int>();
        }
        vector<int> result;
        stack<TreeNode *> treeStack;
        treeStack.push(root);
        while (!treeStack.empty()) {
            TreeNode *temp = treeStack.top();
            result.push_back(temp->val);
            treeStack.pop();
            if (temp->right!=NULL) {
                treeStack.push(temp->right);
            }
            if (temp->left!=NULL) {
                treeStack.push(temp->left);
            }
        }
        return result;
    }
};
```

written by [yulingtianxia](#) original link [here](#)

From [LeetCoder](#).