ttp://www.lintcode.com/problem/binary-tree-preorder-traversal/ (http://www.lintcode.com/problem/binary-tree-preorder-traversal/)

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

or example:
iven binary tree {1,#,2,3},

```
\
 2
 /
3
```

turn [1,2,3].

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

细题解请见九章算法微博：http://weibo.com/3948019741/BzQahxewZ (http://weibo.com/3948019741/BzQahxewZ)

```
/**
 * 本代码由九章算法编辑提供。版权所有，转发请注明出处。
 * - 九章算法致力于帮助更多中国人找到好的工作，教师团队均来自硅谷和国内的一线大公司在职工程师。
 * - 现有的面试培训课程包括：九章算法班，系统设计班，算法强化班，Java入门与基础算法班，Android 项目实战班，Big Data 项目实
战班，
 * - 更多详情请见官方网站：http://www.jiuzhang.com/?source=code
 */

Version 0: Non-Recursion (Recommend)
/**
 * Definition for binary tree
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
public class Solution {
    public List<Integer> preorderTraversal(TreeNode root) {
        Stack<TreeNode> stack = new Stack<TreeNode>();
        List<Integer> preorder = new ArrayList<Integer>();

        if (root == null) {
            return preorder;
        }

        stack.push(root);
        while (!stack.empty()) {
            TreeNode node = stack.pop();
            preorder.add(node.val);
            if (node.right != null) {
                stack.push(node.right);
            }
            if (node.left != null) {
                stack.push(node.left);
            }
        }

        return preorder;
    }
}
```

```java
//Version 1: Traverse
public class Solution {
    public ArrayList<Integer> preorderTraversal(TreeNode root) {
        ArrayList<Integer> result = new ArrayList<Integer>();
        traverse(root, result);
        return result;
    }
    // 把root为跟的preorder加入result里面
    private void traverse(TreeNode root, ArrayList<Integer> result) {
        if (root == null) {
            return;
        }

        result.add(root.val);
        traverse(root.left, result);
        traverse(root.right, result);
    }
}

//Version 2: Divide & Conquer
public class Solution {
    public ArrayList<Integer> preorderTraversal(TreeNode root) {
        ArrayList<Integer> result = new ArrayList<Integer>();
        // null or leaf
        if (root == null) {
            return result;
        }

        // Divide
        ArrayList<Integer> left = preorderTraversal(root.left);
        ArrayList<Integer> right = preorderTraversal(root.right);

        // Conquer
        result.add(root.val);
        result.addAll(left);
        result.addAll(right);
        return result;
    }
}
```