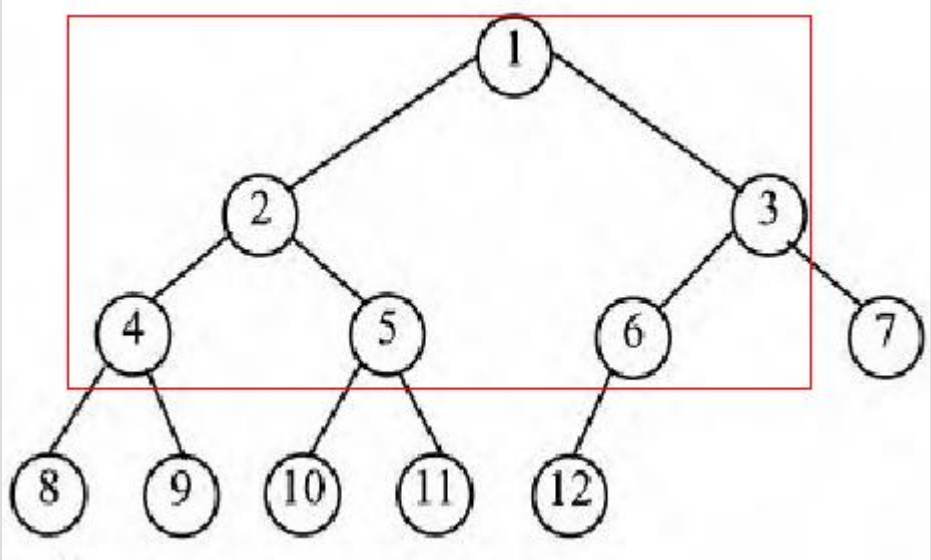
题目描述：

给定一个以顺序储存结构存储整数值的完全二叉树序列（最多 1000 个整数），请找出此完全二叉树的所有非叶子节点部分，然后采用后序遍历方式将此部分树（不包含叶子）输出。

1、只有一个节点的树，此节点认定为根节点（非叶子）。

2、此完全二叉树并非满二叉树，可能存在倒数第二层出现叶子或者无右叶子的情况



其他说明：二叉树的后序遍历是基于根来说的，遍历顺序为：左-右-根

输入描述：

一个通过空格分割的整数序列字符串

输出描述：

非叶子部分树结构的后序遍历结果

补充说明：
输出数字以空格分隔

示例 1

输入：

```
1 2 3 4 5 6 7
```
输出：
```
2 3 1
```
说明：
找到非叶子部分树结构，然后采用后续遍历输出

```
import java.util.*;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {

    static class Node{
        String val;
        Node left,right,parent;
        Node(String val){
            this.val=val;
        }
        Node(String val,Node parent){
            this.val=val;
            this.parent=parent;
        }
    }
    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);
        // 注意 hasNext 和 hasNextLine 的区别
        while (in.hasNext()) { // 注意 while 处理多个 case
            String str = in.nextLine();
            String[] strs = str.split(" ");
            Queue<Node> queue = new LinkedList<>();
            List<Node> list = new ArrayList<>();
            Node root = new Node(strs[0]);
            queue.add(root);
            list.add(root);
            for(int i=1;i<strs.length;i=i+2){
                Node parent = queue.poll();
                Node left = new Node(strs[i],parent);
                parent.left=left;
                queue.add(left);
                list.add(left);
                if(i+1< strs.length){
                    Node right = new Node(strs[i+1],parent);
                    parent.right=right;
                    queue.add(right);
                    list.add(right);
```

```
                }
            }
            for(Node node:list){
                if(node.parent!=null && node.left ==    null && node.right==null){
                    if(node.parent.left == node) node.parent.left = null;
                    if(node.parent.right == node) node.parent.right = null;
                }
            }
            func(root);
        }
    }

    private static void func(Node root) {
        if(root==null) return;
        func(root.left);
        func(root.right);
        System.out.print(root.val+" ");
    }
}
```