https://course.acciojob.com/idle?question=2f52a511-7f24-4df5-8f34-f5b52f50caa4

- EASY
- Max Score: 30 Points

Internal nodes of binary tree

You are given a pointer to the root of a binary tree. You have to print the internal nodes of the binary tree in level order.

NOTE Internal nodes are nodes that have at least 1 child.

You need to complete the given function. The input and printing of output will be handled by the driver code.

Input Format

The first line contains the number of test cases `t`.

The next `t` lines contain a string each, which denotes the binary tree in level order traversal. `N` represents a null node.

Output Format

For each test case return a list of internal nodes.

Example 1

Input

```
1
1 2 3
```

Output

```
1
```

Explanation

Only '1' has children.

Example 2

Input

```
1
1 2 3 N N 4 N
```

Output

```
1 3
```

Explanation

'1' and '3' have children.

Constraints

1 <= T <= 10

1 <= N <= 10000

Topic Tags

- **Trees**
-

My code

// in java

```
import java.util.LinkedList;
import java.util.Queue;
import java.io.*;
import java.util.*;

class Node{
    int data;
    Node left;
    Node right;
    Node(int data){
        this.data = data;
        left=null;
        right=null;
    }
}


class Main {
    static Node buildTree(String str){
        if(str.length()==0 || str.charAt(0)=='N'){
            return null;
        }
        String ip[] = str.split(" ");
        Node root = new Node(Integer.parseInt(ip[0]));
        Queue<Node> queue = new LinkedList<>();
        queue.add(root);
        int i = 1;
        while(queue.size()>0 && i < ip.length) {
            Node currNode = queue.peek();
            queue.remove();
            String currVal = ip[i];
```

```
            if(!currVal.equals("N")) {
                currNode.left = new Node(Integer.parseInt(currVal));
                queue.add(currNode.left);
            }
            i++;
            if(i >= ip.length)
                break;
            currVal = ip[i];
            if(!currVal.equals("N")) {
                currNode.right = new Node(Integer.parseInt(currVal));
                queue.add(currNode.right);
            }
            i++;
        }
        return root;
    }
    void inOrder(Node node) {
        if (node == null) {
            return;
        }
        inOrder(node.left);
        System.out.print(node.data + " ");
        inOrder(node.right);
    }

        public static void main (String[] args) throws IOException{
                BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
                int t=Integer.parseInt(br.readLine());
                while(t-- > 0){
                        String s = br.readLine();
                        Node root = buildTree(s);
```

```
                Solution tree = new Solution();
                ArrayList<Integer> arr = tree.internalNode(root);
                for(int x : arr)
                System.out.print(x +" ");
                System.out.println();
            }
        }
}



class Solution{
    ArrayList<Integer> internalNode(Node root) {
        ArrayList<Integer>a=new  ArrayList<Integer>();
              Queue<Node> q= new  LinkedList<>();
            q.add(root);

                        while(!q.isEmpty())
                            {
                            int f=0;//flag check is child
                                Node t=q.remove();
                                if(t.left!=null)
                                {
                                    q.add(t.left);
                                    f=1;
                                }
                                if(t.right!=null)
                                {
                                    q.add(t.right);
                                    f=1;
                                }
                                if(f!=0)
```

```
                    a.add(t.data);
                }
    return a;
     }
}
```