https://course.acciojob.com/idle?question=486c713a-c48f-455d-8c8a-a36b89e31da1

- EASY
- Max Score: 30 Points

Invert binary tree

You are given a pointer to the `root` of a binary tree. You have to invert the binary tree.

Note 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.

For each test case: You are given a pointer to the root of the binary tree.

Output Format

For each test case, invert the binary tree and return the root.

Example 1

Input

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

Output

```
 1
 / \
3   2
```

Explanation

```
The given tree looks like this:

  1
 / \
2   3

We invert the given binary tree.
```

Example 2

Input

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

Output

```
  1
  / \
 3   2
  \
   4
```

Explanation

```
The given tree looks like this:

  1
 / \
2   3
   /
  4

We invert the given binary tree.
```

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 g = new Solution();
            root =g.invert(root);
            Main mt = new Main();
            mt.inOrder(root);
            System.out.println();
        }
    }
}


class Solution {
    public static Node invert(Node root) {
            if(root==null)
                    return root;
            Node n=root.left;
            root.left=root.right;
            root.right=n;
            invert( root.left);
            invert( root.right);
                return root;

    }
}
```