https://course.acciojob.com/idle?question=2ac5331d-9ff8-450c-b4fd-5e998522513e

- 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 only line of input contains the level order traversal of the input binary tree.**

## Output Format

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

## Example 1

Input

```
1 2 3
```

Output

```
 1
 / \
3   2
```

Explanation

```
The given tree looks like this:

  1
 / \
2   3

We invert the given binary tree.
```

## Example 2

Input

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

Output

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

Explanation

```
The given tree looks like this:

  1
 / \
2   3
   /
  4

We invert the given binary tree.
```

## Constraints

**1 <= num of nodes <= 2000**

**$-10^4$ <= node.val <= $10^4$**

**Topic Tags**

- **Trees**
- **BFS**
- **DFS**
- **Binary Search**

# 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));
        String s = br.readLine();
        Node root = buildTree(s);
        Solution g = new Solution();
        root = g.invert(root);
        Main mt = new Main();
        mt.inOrder(root);
    }
}

class Solution {
 static void inv(Node root)
        {
                if(root==null)
                        return;
                Node t=root.left;
                root.left=root.right;
                root.right=t;
                inv(root.left);
                inv(root.right);
        }
    public Node invert(Node root) {
```

```
        // write code here
        inv(root);
                return root;
    }
}
```