```
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */
public class Codec {

    // Encodes a tree to a single string.
    public String serialize(TreeNode root) {
        if(root == null)    return "N";
        StringBuilder sb = new StringBuilder();
        Queue<TreeNode> q = new LinkedList<>();
        q.offer(root);
        while (!q.isEmpty()) {
            TreeNode node = q.poll();
            if (node == null) {
                sb.append("N,");
            } else {
                sb.append(node.val + ",");
                q.offer(node.left);
                q.offer(node.right);
            }
        }
        sb.deleteCharAt(sb.length()-1);
        return sb.toString();
    }

    // Decodes your encoded data to tree.
    public TreeNode deserialize(String data) {
        String[] strs = data.split("\\,");
        if (strs.length == 1) return null;
        TreeNode root = new TreeNode(Integer.parseInt(strs[0]));
        Queue<TreeNode> q = new LinkedList<TreeNode>();
        q.offer(root);
        int i = 1;
        while (!q.isEmpty()) {
            TreeNode node = q.poll();
            int left = i++, right = i++;
            if (!strs[left].equals("N")) {
                TreeNode l = new TreeNode(Integer.parseInt(strs[left]));
                node.left = l;
                q.offer(l);
            }
            if (!strs[right].equals("N")) {
                TreeNode r = new TreeNode(Integer.parseInt(strs[right]));
                node.right = r;
                q.offer(r);
            }
        }
        return root;
```

```
    }
}

// Your Codec object will be instantiated and called as such:
// Codec codec = new Codec();
// codec.deserialize(codec.serialize(root));




// this works for both BT and BST
/**
 * Definition for a binary tree node.
 * public class TreeNode {
 *     int val;
 *     TreeNode left;
 *     TreeNode right;
 *     TreeNode(int x) { val = x; }
 * }
 */

public class Codec {

    int i = 0;
    // Encodes a tree to a single string.
    public String serialize(TreeNode root) {
        if(root == null){
            return "#";
        }
        return String.valueOf(root.val) + "," + serialize(root.left) + "," + serialize(root.right);
    }

    // Decodes your encoded data to tree.
    public TreeNode deserialize(String data) {
        String[]strs = data.split(",");
        i = 0;
        return deserialize(strs);
    }
    private TreeNode deserialize(String[]arr){
        if(arr[i].equals("#")){
            i++;
            return null;
        }
        TreeNode root = new TreeNode(Integer.parseInt(arr[i++]));
        root.left = deserialize(arr);
        root.right = deserialize(arr);
        return root;
    }
}
/*
public class Codec {

    // Encodes a tree to a single string.
```

```
    public String serialize(TreeNode root) {
        if(root == null)    return "N";
        StringBuilder sb = new StringBuilder();
        Queue<TreeNode> q = new LinkedList<>();
        q.offer(root);
        while (!q.isEmpty()) {
            TreeNode node = q.poll();
            if (node == null) {
                sb.append("N,");
            } else {
                sb.append(node.val + ",");
                q.offer(node.left);
                q.offer(node.right);
            }
        }
        sb.deleteCharAt(sb.length()-1);
        return sb.toString();
    }

    // Decodes your encoded data to tree.
    public TreeNode deserialize(String data) {
        String[] strs = data.split("\\,");
        if (strs.length == 1) return null;
        TreeNode root = new TreeNode(Integer.parseInt(strs[0]));
        Queue<TreeNode> q = new LinkedList<TreeNode>();
        q.offer(root);
        int i = 1;
        while (!q.isEmpty()) {
            TreeNode node = q.poll();
            int left = i++, right = i++;
            if (!strs[left].equals("N")) {
                TreeNode l = new TreeNode(Integer.parseInt(strs[left]));
                node.left = l;
                q.offer(l);
            }
            if (!strs[right].equals("N")) {
                TreeNode r = new TreeNode(Integer.parseInt(strs[right]));
                node.right = r;
                q.offer(r);
            }
        }
        return root;
    }
}
*/

// Your Codec object will be instantiated and called as such:
// Codec codec = new Codec();
// codec.deserialize(codec.serialize(root));
```