https://course.acciojob.com/idle?question=a517da49-60d0-4a1b-9abd-39c38eae8db3

- EASY
- Max Score: 30 Points

Root Leaf Path Sum

You are given a pointer to the `head` of the binary tree. You have to find the sum of all the numbers which are formed from root to leaf paths.

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 `head` of the binary tree.

Output Format

For each test case return an integer denoting the sum of all numbers from root to leaf.

Example 1

Input

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

Output

```
25
```

Explanation

The given tree is

```
    1
   /   \
  2     3
```

The number formed in the tree are 12 and 13. Therefore sum is 25.

Example 2

Input

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

Output

```
137
```

Explanation

The given tree is

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

The number formed in the tree are 124 and 13. Therefore sum is 137.

Constraints

1 <= T <= 10

1 <= N <= 12

1 <= A[i] <= 9

Topic Tags

- **Trees**

My code

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

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;
    }
    static void printInorder(Node root)
    {
        if(root == null)
            return;

        printInorder(root.left);
        System.out.print(root.data+" ");

        printInorder(root.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();
                System.out.println(g.treePathsSum(root));
            }
        }
}
```

```
class Solution{
	static long fun(Node root,long n)
	{
		if(root.left==null && root.right==null)
			return n+root.data;
		if(root.left==null && root.right!=null)
		{
			n=n+root.data;
			return fun(root.right,n*10 );
		}
		  if(root.left!=null && root.right==null)
		{
			n=n+root.data;
			return fun(root.left,n*10 );
		}
		n=n+root.data;
		long a=fun(root.left,n*10);
		long b=fun(root.right,n*10 );
		return a+b;
	}
	public long treePathsSum(Node root){
		//Write code here
		  long pass=0;
		return fun(root,pass);

	}
}
```