### Lowest Common Ancestor in a BST

Given a Binary Search Tree (with all values unique) and two node values n1 and n2 (n1!=n2). Find the Lowest Common Ancestors of the two nodes in the BST.

**Example 1:**

**Input:**

              5

           /  \

          4      6

    /       \

   3       7

                 \

                    8

n1 = 7, n2 = 8

**Output:** 7

**Example 2:**

**Input:**

2

  / \

  1 3

n1 = 1, n2 = 3

**Output:** 2

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **LCA()** which takes the root Node of the BST and two integer values n1 and n2 as inputs and returns the Lowest Common Ancestor of the Nodes with values n1 and n2 in the given BST.

**Expected Time Complexity:** O(Height of the BST).  
**Expected Auxiliary Space:** O(Height of the BST).

**Constraints:**  
1 <= N <= 104

**Company Tags**

[**Flipkart**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Flipkart) [**Accolite**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Accolite) [**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft) [**Samsung**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Samsung) [**MAQ Software**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=MAQ%20Software) [**Synopsys**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Synopsys)

//{ Driver Code Starts

//Initial Template for 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 CodingMaxima {

static Node buildTree(String str){

if(str.length()==0 || str.charAt(0)=='N'){

return null;

}

String ip[] = str.split(" ");

// Create the root of the tree

Node root = new Node(Integer.parseInt(ip[0]));

// Push the root to the queue

Queue<Node> queue = new LinkedList<>();

queue.add(root);

// Starting from the second element

int i = 1;

while(queue.size()>0 && i < ip.length) {

// Get and remove the front of the queue

Node currNode = queue.peek();

queue.remove();

// Get the current node's value from the string

String currVal = ip[i];

// If the left child is not null

if(!currVal.equals("N")) {

// Create the left child for the current node

currNode.left = new Node(Integer.parseInt(currVal));

// Push it to the queue

queue.add(currNode.left);

}

// For the right child

i++;

if(i >= ip.length)

break;

currVal = ip[i];

// If the right child is not null

if(!currVal.equals("N")) {

// Create the right child for the current node

currNode.right = new Node(Integer.parseInt(currVal));

// Push it to the queue

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 Exception{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int t=Integer.parseInt(br.readLine());

//Scanner sc = new Scanner(System.in);

while(t > 0){

String s = br.readLine();

Node root = buildTree(s);

BST g = new BST();

String X = br.readLine();

String arr[] = X.split(" ");

int x , y;

x = Integer.parseInt(arr[0]);

y = Integer.parseInt(arr[1]);

System.out.println(g.LCA(root,x,y).data);

t--;

}

}

}

// } Driver Code Ends

//User function Template for Java

class BST

{

//Function to find the lowest common ancestor in a BST.

Node LCA(Node root, int n1, int n2)

{

if(root.data > n1 && root.data > n2){

return LCA(root.left,n1,n2);

}

if(root.data < n1 && root.data < n2){

return LCA(root.right,n1,n2);

}

return root;

}

}