**Check for BST**

Given the root of a binary tree. Check whether it is a BST or not.  
**Note:**We are considering that BSTs can not contain duplicate Nodes.  
A **BST** is defined as follows:

* The left subtree of a node contains only nodes with keys **less than** the node's key.
* The right subtree of a node contains only nodes with keys **greater than** the node's key.
* Both the left and right subtrees must also be binary search trees.

**Example 1:**

**Input:**

   2

/    \

1      3

**Output:** 1

**Explanation:**

The left subtree of root node contains node

with key lesser than the root nodes key and

the right subtree of root node contains node

with key greater than the root nodes key.

Hence, the tree is a BST.

**Example 2:**

**Input:**

2

  \

  7

  \

  6

  \

  5

  \

  9

  \

  2

  \

  6

**Output:** 0

**Explanation:**

Since the node with value 7 has right subtree

nodes with keys less than 7, this is not a BST.

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

**Constraints:**  
0 <= Number of edges <= 100000

**Company Tags**

[**VMWare**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=VMWare) [**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) [**OYO Rooms**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=OYO%20Rooms) [**Samsung**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Samsung)[**Snapdeal**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Snapdeal) [**FactSet**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=FactSet)[**Hike**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Hike)[**MakeMyTrip**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=MakeMyTrip) [**Ola Cabs**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Ola%20Cabs) [**Walmart**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Walmart) [**Goldman Sachs**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Goldman%20Sachs) [**MAQ Software**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=MAQ%20Software) [**Adobe**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Adobe) [**Linkedin**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Linkedin) [**Qualcomm**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Qualcomm) [**Boomerang Commerce**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Boomerang%20Commerce) [**Grey Orange**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=GreyOrange) [**Wooker**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Wooker)

**Topic Tags**

[**Binary Search Tree**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Binary%20Search%20Tree)[**Tree**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Tree) [**Data Structures**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Data%20Structures)

//{ 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 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();

if(g.isBST(root))

System.out.println(1);

else

System.out.println(0);

t--;

}

}

}

// } Driver Code Ends

//User function Template for Java

public class Solution

{

public void inorder(Node root, ArrayList<Integer> arr){

if(root==null)

return ;

inorder(root.left,arr);

arr.add(root.data);

inorder(root.right,arr);

}

boolean isBST(Node root)

{

// if(root==null)

// return true;

ArrayList<Integer> arr=new ArrayList<>();

inorder(root, arr);

return sort(arr);

}

private static boolean sort(ArrayList<Integer> arr){

for (int i = 0; i < arr.size() - 1; i++) {

if (arr.get(i) >= arr.get(i + 1))

return false;

}

return true;

}

}