'''

Mansood Ali participated in a treasure-hunt conatins N boxes in it.

Each box is numbered from 0,1,2,3,...,N-1.

All the boxes are intially locked with the passcodes, except Box-0.

The passcode is written in a envolope and marked with its concern box number.

Each box in the treasure hunt contains a list of envelopes, which has

passcodes to open the other boxes. He can open the boxes in any order,

but he has to start from box-0 intially.

The rule to win the treasure-hunt is to open all the boxes.

Your task is to find out, whether Mansood Ali win the treasure hunt or not.

If he win, print "Win", Otherwise, print "Lost"

Input Format:

-------------

Line-1: An integer, number of boxes.

Next N lines: space separated integers, box numbers.

Output Format:

--------------

Print a string value, Win or Lost

Sample Input-1:

---------------

5

1

2

3

4

3

Sample Output-1:

----------------

Win

Sample Input-2:

---------------

4

1 3

3 0 1

2

0

Sample Output-2:

----------------

Lost

-----------------------------

Write your python code below

-----------------------------

'''

l=[]

n=int(input())

for i in range(n):

l.append(list(map(int,input().split())))

vis=[False for i in range(0,n)]

def dfs(l,src,vis):

vis[src]=True

for i in l[src]:

if (vis[i]==False):

dfs(l,i,vis)

dfs(l,0,vis)

flag=False

# print(vis)

for i in vis:

if(i==False):

flag=True

if(flag):

print("Lost")

else:

    print("Win")

Venkat is a farmer, and he started harvesting the watermelon crop,

the crop grown very well. There are several watermelons in the crop.

Venkat started picking up the watermelons one by one.

After each pick, he keeps the watermelon in a truck placed inside the crop.

The crop is in the from of 2D grid of size m\*n.

You will be given the positions of the truck, Venkat's, and the watermelons.

Positions are represented by the cells in the 2D grid.

Your task is to find the minimum distance for Venkat to collect

all the watermelons and put them inside the truck one by one.

Venkat can only take at most one watermelon at one time

and can move in four directions - up, down, left and right, to the adjacent cell.

The distance is represented by the number of moves.

Input Format:

-------------

Line-1: Two space separated integers m and n, size of crop.

Line-2: Two space separated integers, position of the truck.

Line-3: Two space separated integers, position of Venkat.

Line-4: An integer W, number of watermelons in the crop.

Next W lines: Two space separated integers, positions of watermelon.

Output Format:

--------------

An integer, minimum distance covered by Venkat to pickup all the watermelons

Sample Input-1:

---------------

5 7 //size of the crop as 2D grid.

2 2 //Truck Position

4 4 //Venkat Position

2 //Number of watermelons

3 0 //Watermelon positions

2 5

Sample Output-1:

----------------

12

Sample Input-2:

---------------

5 7

2 2

4 4

5

1 4

2 6

3 0

2 5

4 5

Sample Output-2:

----------------

32

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int m=sc.nextInt();

int n=sc.nextInt();

int t1=sc.nextInt();

int t2=sc.nextInt();

int v1=sc.nextInt();

int v2=sc.nextInt();

int nw=sc.nextInt();

int[][] arr=new int[nw][2];

int min=Integer.MAX\_VALUE;

int nd=0;

for(int i=0;i<arr.length;i++){

int a1=sc.nextInt();

int a2=sc.nextInt();

int wt=Math.abs(a1-t1)+Math.abs(a2-t2);

int vw=Math.abs(a1-v1)+Math.abs(a2-v2);

nd+=wt\*2;

min=Math.min(min,vw-wt);

}

System.out.println(nd+min);

    }

}

Hint:

The expression vw-wt is not wrapped in abs because it is calculating the difference in distances between Venkat's position and the watermelon (vw) and the distance between the truck and the watermelon (wt).

Here's why abs is not used in vw - wt:

- If vw (the distance from Venkat to the watermelon) is greater than wt (the distance from the truck to the watermelon), it means Venkat should carry the watermelon to the truck, so vw - wt will be a positive value.

- If wt is greater than vw, it means it's more efficient for Venkat to carry the watermelon directly to the truck without Venkat picking it up first. In this case, vw - wt will be a negative value.

- If both distances are equal, vw - wt will be 0, indicating there's no additional benefit or cost.

Using abs would remove the distinction between these cases, and it's important for the algorithm to consider whether it's more efficient for Venkat to pick up the watermelon or not.

Therefore, abs is not used in vw - wt to maintain the information about whether carrying the watermelon to the truck is more efficient (positive value) or less efficient (negative value) compared to Venkat picking up the watermelon.

There are N constructions in pyramid structures, you are given the N pyramids

information as an array pyramids[], where pyramid[i]={center\_of\_base, peak} of

the pyramid. A pyramid is described as a right-angled isosceles triangle,

with its center of the base along the x-axis and a right angle at its peak.

More formally, the gradients of ascending and descending the pyramid are 1

and -1 respectively.

A pyramid is considered visible if its peak does not lie within another pyramid

(including the border of other pyramids).

Your task is to return the number of visible pyramids.

Input Format:

-------------

Line-1: An integer N, number of pyramids

Next N lines: Two space separated integers, center\_of\_base and peak.

Output Format:

--------------

Print an integer result.

Sample Input-1:

---------------

3

2 2

6 3

5 4

Sample Output-1:

----------------

2

Explanation:

------------

Look in the hint.

Sample Input-2:

---------------

2

2 4

2 4

Sample Output-2:

----------------

0

Explanation:

------------

The structures on overlap on each other, so we will return 0.

Only 85% test cases passed

import java.util.\*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int[][] arr=new int[n][2];

for(int i=0;i<n;i++){

arr[i][0]=sc.nextInt();

arr[i][1]=sc.nextInt();

}

int[][] base=new int[n][2];

for(int i=0;i<arr.length;i++){

base[i][0]=arr[i][0]-arr[i][1];

base[i][1]=arr[i][0]+arr[i][1];

}

Arrays.sort(base,new Comparator<int[]>(){

public int compare(final int[] e1,final int[] e2){

final int x1=e1[0];

final int x2=e2[0];

return Integer.compare(x1,x2);

}

});

int count=0;

for(int i=0;i<n-1;i++){

if(base[i][0]==base[i+1][0]){

continue;

}

if(arr[i][1]>arr[i+1][1]){

if(base[i+1][1]>base[i][1]){

count+=1;

}

else{

continue;

}

}

count+=1;

}

System.out.println(count);

}

}