'''

Clavius, the person who suggested grouping of data using brackets.

He has suggested that the grouping should be well formed.

Well formed groupings are as follows, (), [], {}, {()}, {[]()}, {[]}(), etc.

You will be given a string S, return true if the string S is a well formed

grouping, otherwise false.

Note: S contains only characters ( ) [ ] { }

Input Format:

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

A string S, contains only characters (){}[]

Output Format:

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

Print a boolean value.

Sample Input-1:

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

{[([])[]]}

Sample Output-1:

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

true

Sample Input-2:

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

{[([])[]}]

Sample Output-2:

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

false

Write your python code below

'''

n=input()

count=0

l=[]

k=0

for i in n:

if(i=='[' or i=='(' or i=='{'):

l.append(i)

elif(i==']'):

if(l.pop()=='['):

continue

else:

k=1

print('false')

break

elif(i==')'):

if(l.pop()=='('):

continue

else:

k=1

print('false')

break

elif(i=='}'):

if(l.pop()=='{'):

continue

else:

k=1

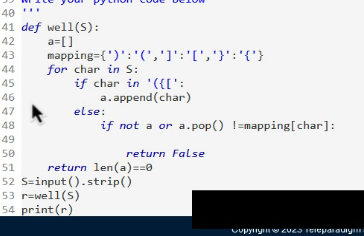
print('false')

break

# print(l)

if(k==0):

print('true')



Every kid enjoys getting out of the classroom and exploring, and

school excursions are one of the finest ways to do so.

ABC School organised a field trip for its students. Students are requested

to form a line, and their heights are supplied to you. You need to make

the students to stand in ascending order of their heights.

Your task is to find out, the minimum number of students who must change

their positions in order to make every student to stand in ascending order.

Note: When a student is selected, the student can change position in a

possible way among all of them, and

the other students remain in their positions.

Input Format:

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

Line-1: An integer N, number of the students.

Line-2: N space separated integers, heights of the students.

Output Format:

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

Print an integer, minimum number of moves.

Sample Input-1:

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

6

1 1 4 2 1 3

Sample Output-1:

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

3

Explanation:

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

Initially Heights are as follows : [1,1,4,2,1,3]

Targeted Heights are as follows : [1,1,1,2,3,4]

On index 2: you have 4 vs 1 so you have to change the position of student.

On index 4: you have 1 vs 3 so you have to change the position of student.

On index 5: you have 3 vs 4 so you have to change the position of student.

Sample Input-2:

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

5

5 1 4 2 3

Sample Output-2:

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

5

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];

int[] arr1=new int[n];

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

arr[i]=sc.nextInt();

arr1[i]=arr[i];

}

Arrays.sort(arr1);

int count=0;

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

if(arr[i]!=arr1[i]){

count+=1;

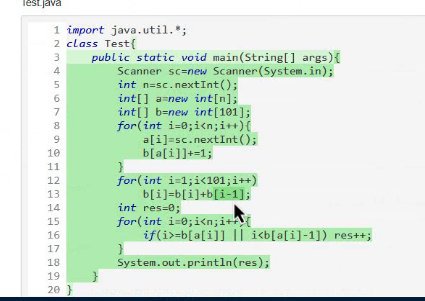
}

}

System.out.println(count);

}

}





MotorSport Ltd hosting a Racing Championship. Mr. Ajith is participating

in car races. Each race start and end in perticular time intervals.

You are given an array of racing time intervals consisting of

start and end times [[s1,e1],[s2,e2],...] (s < e ) of N races, in which

Ajith has to participate. Your task is to determine whether Ajith can

in all the races or not.

NOTE: If a race starts at time 'a' ends at time 'b',

another race can start at 'b'.

Input Format:

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

Line-1: An integer N, number of races Ajith has to participate.

Next N lines: Two space separated integers, start and end time of each race.

Output Format:

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

Print a boolean value.

Sample Input-1:

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

3

0 30

5 10

15 20

Sample Output-1:

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

false

Sample Input-2:

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

3

0 10

15 25

30 35

Sample Output-2:

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

True

1 test case didn’t pass (8th test case)

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();

}

boolean flag=true;

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

if(arr[i-1][1]>arr[i][0]){

flag=false;

break;

}

}

System.out.println(flag);

}

}

