Distributed Online Round

A. Testrun

B. almost_sorted

C. mutexes

D. johnny

E. highest_mountain

Contest Analysis

Questions asked 6

Submissions

Testrun	
0pt	Not attempted **0/64 users** correct (0%)
almost_sorted	
1pt	Not attempted **194/203 users** correct (96%)
7pt	Not attempted **104/187 users** correct (56%)
mutexes	
2pt	Not attempted **84/147 users** correct (57%)
20pt	Not attempted **48/69 users** correct (70%)
johnny	
2pt	Not attempted **91/105 users** correct (87%)
30pt	Not attempted **17/70 users** correct (24%)
highest_mountain	
1pt	Not attempted **43/61 users** correct (70%)
37pt	Not attempted **0/9 users** correct (0%)

Top Scores

mk.al13n	63
ecnerwala	63
shik	63
Marcin.Smulewicz	56
WJMZBMR	56
bmerry	43
ZbanIlya	43
wan92hy	42
simonlindholm	42
dreamoon	42

Problem E. highest_mountain

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

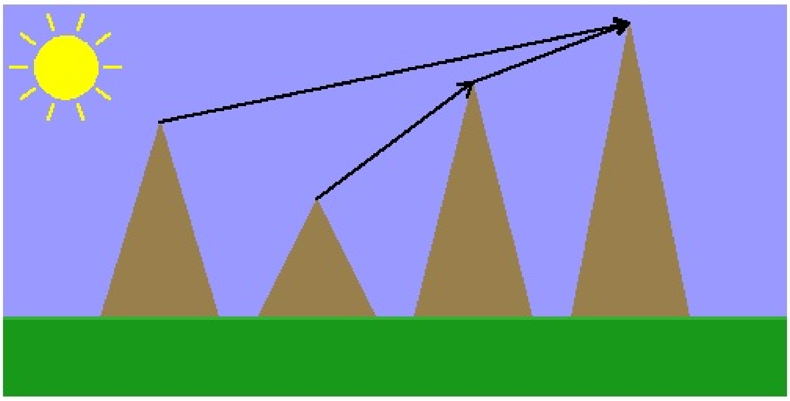
small 1 points *2 minute timeout*	The contest is finished.
large 37 points *10 minute timeout*	The contest is finished.

Don't know what distributed problems are about? See our guide.

Problem

You were born and live in a small town in a remote mountain range extending east to west. In this mountain range there is a peak every kilometer, and there are no intermediate peaks. Recently, you checked on the Internet what the highest peak in the range is, and were suprised — from your town a different peak seems to be the highest. And when you went for a walk to the nearest mountain top, yet another peak appeared to be highest. You're not sure the Internet data is correct (they write all sorts of stuff on the Internet!), so you decided to compile your own list of potentially highest peaks in the range.

Your list will contain every peak **B** with the following property: if **B** is visible from some other peak **A**, no peak beyond **B** is visible from **A**. Formally, this means that if **B** lies, say, to the east of **A**, then all peaks between **A** and **B** are below the line connecting **A** and **B**, and all the peaks to the east of **B** are below or on that line.



In this example, the fourth peak is the last one you see to the east from the first and third peaks. From the second peak, to the east you only see the third peak. Your list will include only peaks 1 and 4: peak 2 is visible from peak 3, but is not the farthest (1 is), and peak 3 is visible from peak 1, but is not the farthest (4 is).

The rationale for this criterion is that you figure that if from some peak **A** you can see peak **C**, and you can also see some other peak **D** that lies in the same direction and is more distant, then **C** is'obviously not the highest peak in the range (because either **A**, or **D**, is higher). You don't trust your intuition any more, so even if the highest visible peak in any direction appears to be much lower than the one you're standing on (for instance, you are standing on a peak of height 1000, and the next and last peak to the east is of height 1), you will consider the peak of height 1 to be a candidate for your list.

Input

The input library will be called "highest_mountain"; see the sample inputs below for examples in your language. It will define two methods: NumberOfPeaks(), which will return the number of peaks in the range, and GetHeight(i), which will return the height of the *i*th peak from the west, for 0 ≤ i < NumberOfPeaks().
One call to GetHeight will take approximately 0.1 microseconds.

Output

Output one number: the total number of the peaks that will be included in your list.

Limits

Each node will have access to 128MB of RAM, and a time limit of 6 seconds.
0 ≤ GetHeight(i) ≤ 10^9 for all i with 0 ≤ i < NumberOfPeaks().

Small input

Your solution will run on 10 nodes.
1 ≤ NumberOfPeaks() ≤ 1000.

Large input

Your solution will run on 100 nodes.
1 ≤ NumberOfPeaks() ≤ 4 × 10^8.

Sample

Input	Output

`See sample input files below.`	`For sample input 1:`
	`2`
	`For sample input 2:`
	`3`
	`For sample input 3:`
	`4`

Sample input libraries:
Sample input for test 1: highest_mountain.h [CPP] highest_mountain.java [Java]
Sample input for test 2: highest_mountain.h [CPP] highest_mountain.java [Java]
Sample input for test 3: highest_mountain.h [CPP] highest_mountain.java [Java]