G2. Ruler (hard version)

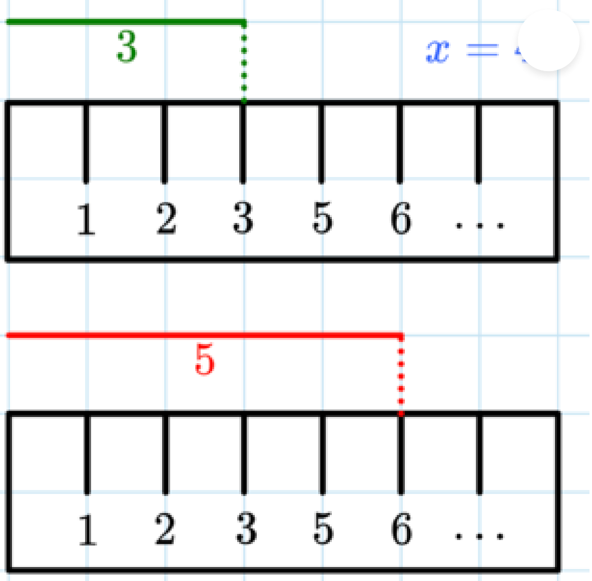
time limit per test: 1 second
memory limit per test: 256 megabytes

This is the hard version of the problem. The only difference between the two versions is that in this version, you can make at most $\mathbf{7}$ queries.

This is an interactive problem. If you are unsure how interactive problems work, then it is recommended to read the guide for participants.

We have a secret ruler that is missing one number x $(2 \leq x \leq 999)$. When you measure an object of length y, the ruler reports the following values:

- If $y < x$, the ruler (correctly) measures the object as having length y.
- If $y \geq x$, the ruler incorrectly measures the object as having length $y+1$.



The ruler above is missing the number 4, so it correctly measures the first segment as length 3 but incorrectly measures the second segment as length 6 even though it is actually 5.

You need to find the value of x. To do that, you can make queries of the following form:

- **?** a b — in response, we will measure the side lengths of an $a \times b$ rectangle with our ruler and multiply the results, reporting the measured area of the rectangle back to you. For example, if $x = 4$ and you query a 3×5 rectangle, we will measure its side lengths as 3×6 and report 18 back to you.

Find the value of x. You can ask at most $\mathbf{7}$ queries.

Input

Each test contains multiple test cases. The first line of input contains a single integer t $(1 \leq t \leq 1000)$ — the number of test cases.

Interaction

There is no initial input for each test case. You should begin the interaction by asking a query.

To make a query, output a single line of the form **?** a b $(1 \leq a, b \leq 1000)$. In response, you will be told the measured area of the rectangle, according to our secret ruler.

When you are ready to print the answer, output a single line of the form **!** x $(2 \leq x \leq 999)$. After that, proceed to process the next test case or terminate the program if it was the last test case. Printing the answer does not count as a query.

The interactor is **not** adaptive, meaning that the answer is known before the participant asks the queries and doesn't depend on the queries asked by the participant.

If your program makes more than 7 queries for one set of input data, makes an invalid query, or prints the wrong value of x, then **the response to the query will be** -1. After receiving such a response, your program should immediately terminate to receive the verdict `Wrong Answer`. Otherwise, you can get an arbitrary verdict because your solution will continue to read from a closed stream.

After printing a query do not forget to output the end of line and flush the output. Otherwise, you may get `Idleness limit exceeded` verdict. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see the documentation for other languages.

Codeforces Round 964 (Div. 4)

Finished

Practice

☆

→ Virtual participation

Virtual contest is a way to take part in past contest, as close as possible to participation on time. It is supported only ICPC mode for virtual contests. If you've seen these problems, a virtual contest is not for you - solve these problems in the archive. If you just want to solve some problem from a contest, a virtual contest is not for you - solve this problem in the archive. Never use someone else's code, read the tutorials or communicate with other person during a virtual contest.

Start virtual contest

→ Clone Contest to Mashup

You can clone this contest to a mashup.

Clone Contest

→ Submit?

Language: GNU G++20 13.2 (64 bit, win

Choose file: Choose File No file chosen

Submit

→ Last submissions

Submission	Time	Verdict
275152938	Aug/08/2024 01:43	Accepted
275152865	Aug/08/2024 01:42	Time limit exceeded on test 1
275152593	Aug/08/2024 01:36	Time limit exceeded on test 1
275152361	Aug/08/2024 01:32	Time limit exceeded on test 1

→ Problem tags

binary search interactive
ternary search *1700

No tag edit access

→ Contest materials

- Announcement (en)
- Video Tutorial (en)
- Tutorial #2 (en)

Hacks

To make a hack, use the following format.

The first line should contain a single integer t $(1 \le t \le 1000)$ — the number of test cases.

The only line of each test case should contain a single integer x $(2 \le x \le 999)$ — the missing number on the ruler.

Example

input Copy

```
2

18

25


9999
```

output Copy

```
? 3 5

? 4 4

! 4
? 99 100

! 100
```

Note

In the first test, the interaction proceeds as follows.

Solution	Jury	Explanation
	2	There are 2 test cases.
`? 3 5`	18	Secretly, the jury picked $x = 4$. The solution requests the 3×5 rectangle, and the jury responds with $3 \times 6 = 18$, as described in the statement.
`? 4 4`	25	The solution requests the 4×4 rectangle, which the jury measures as 5×5 and responds with 25.
`! 4`		The solution has somehow determined that $x = 4$, and outputs it. Since the output is correct, the jury continues to the next test case.
`? 99 100`	1	Secretly, the jury picked $x = 100$. The solution requests the 99×100 rectangle, which the jury measures as 99×101 and responds with 9999.
`! 100`		The solution has somehow determined that $x = 100$, and outputs it. Since the output is correct and there are no more test cases, the jury and the solution exit.

Note that the line breaks in the example input and output are for the sake of clarity, and do not occur in the real interaction.