C3. Hacking Numbers (Hard Version)

time limit per test: 2 seconds
memory limit per test: 256 megabytes

This is the hard version of the problem. In this version, the limit of commands you can send is described in the statement. You can make hacks only if all versions of the problem are solved.

This is an interactive problem.

Welcome, Duelists! In this interactive challenge, there is an unknown integer x $(1 \le x \le 10^9)$. You must make it equal to a given integer in the input n. By harnessing the power of "Mathmech" monsters, you can send a command to do one of the following:

Command	Constraint	Result	Case	Update	Jury's response
"add y"	$-10^{18} \le y \le 10^{18}$	$\text{res} = x + y$	if $1 \le \text{res} \le 10^{18}$	$x \leftarrow \text{res}$	"1"
			else	$x \leftarrow x$	"0"
"mul y"	$1 \le y \le 10^{18}$	$\text{res} = x \cdot y$	if $1 \le \text{res} \le 10^{18}$	$x \leftarrow \text{res}$	"1"
			else	$x \leftarrow x$	"0"
"div y"	$1 \le y \le 10^{18}$	$\text{res} = x / y$	if y divides x	$x \leftarrow \text{res}$	"1"
			else	$x \leftarrow x$	"0"
"digit"	—	$\text{res} = S(x)$ *	—	$x \leftarrow \text{res}$	"1"

Let $f(n)$ be the minimum integer such that there is a sequence of $f(n)$ commands that transforms x into n for all $x(1 \le x \le 10^9)$. You do not know the value of x in advance. Find $f(n)$ such that, no matter what x is, you can always transform it into n using at most $f(n)$ commands.

Your task is to change x into n using **at most** $f(n)$ commands.

* $S(n)$ is a function that returns the sum of all the individual digits of a non-negative integer n. For example, $S(123) = 1 + 2 + 3 = 6$

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 5000$). The description of the test cases follows.

The first and only line of each test case contains one integer n $(1 \le n \le 10^9)$.

Interaction

The interaction for each test case begins by reading the integer n.

To send a command, output a line in the following format:

- "add y" Add some integer y $(-10^{18} \le y \le 10^{18})$ to x.
  The jury will output "1" if $x + y$ is within $[1, 10^{18}]$ (**successful**), and "0" otherwise. If successful, update $x \leftarrow x + y$.
- "mul y" Multiply x by a positive integer y $(1 \le y \le 10^{18})$.
  The jury will output "1" if $x \cdot y$ is within $[1, 10^{18}]$ (**successful**), and "0" otherwise. If successful, update $x \leftarrow x \cdot y$.
- "div y" Divide x by a positive integer y $(1 \le y \le 10^{18})$.
  The jury will output "1" if y is a divisor of x (**successful**), and "0" otherwise. If successful, update $x \leftarrow \frac{x}{y}$.

Codeforces Round 1025 (Div. 2)

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++23 14.2 (64 bit, ms

Choose file: Choose File No file chosen

Submit

→ Last submissions

Submission	Time	Verdict
322052131	May/30/2025 15:32	Accepted
322051713	May/30/2025 15:28	Wrong answer on test 2
322051411	May/30/2025 15:25	Wrong answer on test 2
322051060	May/30/2025 15:22	Wrong answer on test 2
322050499	May/30/2025 15:18	Idleness limit exceeded on test 2
322050020	May/30/2025 15:13	Wrong answer on test 1

→ Problem tags

constructive algorithms interactive math number theory *2600

No tag edit access

→ Contest materials

- Announcement (en)
- Tutorial (en)

- "`digit`" Make x equal to the sum of its digits.
  The jury will always output "`1`" and update $x \leftarrow S(x)$.

Note that commands are **case sensitive**.

When you have determined that x is equal to n, output a line in the following format:

- "`!`" — where the jury will output a "`1`" if n is equal to x, and "`-1`" otherwise.

Note that answering **does not count** toward your limit of commands.

If your program makes more than $f(n)$ commands ($f(n)$ is described above) for one test case, or makes an invalid command, then the response to the command will be "`-1`". After receiving such a response, your program should immediately terminate to receive the verdict **Wrong Answer**. Otherwise, it may receive any other verdict.

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

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `sys.stdout.flush()` in Python;
- `std::io::stdout().flush()` in Rust;
- see the documentation for other languages.

The interactor is **non-adaptive**. The unknown integer x **does not change** during the interaction.

Hacks

To hack, use the following format.

The first line should contain a single integer t $(1 \le t \le 5000)$ — the number of test cases.

The first line of each test case should contain two positive integers n and x $(1 \le n, x \le 10^9)$ — denoting the unknown integer and the target value to which it should be made equal, respectively.

Example

input

```
2
100

0

1

1

1

5

1

1

1
```

output

```

add -10

add 1

mul 10

!

digit

div 2

!
```

Note

Solution	Jury	Explanation
	2	There are 2 test cases.

	100	In the first test case, the unknown integer $x = 9$ and we have to make it equal to $n = 100$.
`add -10`	0	The answer to "`add -10`" is "`0`". This means that the addition command was not successful as $x + y = 9 + (-10) \leq 0$, and x remains 9 after the command
`add 1`	1	The answer to "`add 1`" is "`1`". This means that the addition command was successful as $x + y = 9 + 1 = 10$, and x changes to 10 after the command.
`mul 10`	1	The answer to "`mul 10`" is "`1`". This means that the multiplication command was successful as $x \cdot y = 10 \cdot 10 = 100$, and x changes to 100 after the command.
`!`	1	The answer to "`!`" is "`1`". This means you have determined that x equals n.
	5	In the second test case, the unknown integer $x = 1234$ and we have to make it equal to $n = 5$.
`digit`	1	The answer to "`digit`" is "`1`". This means that x turned into the sum of its digits $1 + 2 + 3 + 4 = 10$, and x changes to 10 after the command.
`div 2`	1	The answer to "`div 2`" is "`1`". This means that the division command was successful as $y = 2$ is a divisor of $x = 10$, and x changes to $\frac{x}{y} = \frac{10}{2} = 5$ after the command.
`!`	1	The answer to "`!`" is "`1`". This means you have determined that x equals n.

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