Key Recovery

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Did you know? During the selection of ISIJ contestants, ISTJs and ISFJs don't get extra points.

—Mixed-chicken

This is an interactive problem.

Kevin is dissatisfied with his MBTI, so he asked Doraemon for a magical machine: the MBTI converter.

The MBTI converter can convert the MBTI attributes of 8 people at a time. It will generate a "random" new MBTI for all individuals based on their initial MBTIs and a 32-bit key k.

Kevin carefully studied this machine and found that the "random" MBTI it generates actually follows a certain pattern. The machine mainly includes three basic structures that shuffle the MBTI: "`xor`", "`perm`", and "`mix`". Each of these structures takes the MBTIs of 8 individuals as input and outputs 8 MBTIs as a result. In each structure, the input 8 MBTIs are denoted as $in_0, in_1, in_2, \cdots, in_7$ and the output 8 MBTIs are denoted as $out_0, out_1, out_2, \cdots, out_7$. For convenience, Kevin replaces each MBTI with a hexadecimal number according to the following table.

INFP	INFJ	INTP	INTJ	ISFP	ISFJ	ISTP	ISTJ
0	1	2	3	4	5	6	7
ENFP	ENFJ	ENTP	ENTJ	ESFP	ESFJ	ESTP	ESTJ
8	9	a	b	c	d	e	f

"`xor`" Structure:

The `xor` structure flips certain attributes of the MBTI based on the key k.

Let k_j be the j-th binary bit of the key k ($0 \leq j < 32$). The functionality of the `xor` structure can be expressed as:

$$out_i = in_i \oplus (\overline{k_{4i+3}k_{4i+2}k_{4i+1}k_{4i}})_2$$

where $\oplus$ denotes the bitwise XOR[1].

"`perm`" Structure:

The `perm` structure replaces the MBTI, where each MBTI is replaced by another fixed MBTI.

The functionality of the `perm` structure can be expressed as:

$$out_i = p_{in_i}$$

where p is a fixed array of length 16 with the following values:

p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}	p_{14}	p_{15}
7	0	5	6	b	f	4	3	1	d	c	e	8	9	a	2

"`mix`" Structure:

The `mix` structure is used to mix the MBTIs of different people, with each output determined by two inputs.

The `mix` structure requires an intermediate array t of length 8, defined as:

$$t_i = \begin{cases} in_i \ll 1 & 0 \le in_i < 8 \\ ((in_i \oplus 8) \ll 1) \oplus 3 & 8 \le in_i < 16 \end{cases}$$

where $\oplus$ denotes the bitwise XOR[1] and $\ll$ denotes logical left shift of four binary digits[2].

The functionality of the `mix` structure can be expressed as:

$$out_i = in_{i \ggg 1} \oplus (t_{(i \ggg 1) \oplus 1})$$

where $\oplus$ denotes the bitwise XOR[1], and $\ggg$ denotes circular right shift of three binary digits[3].

The initial 8 MBTIs will sequentially go through 19 structures to produce the final output of 8 MBTIs. These structures are: `xor, perm, mix, xor, perm, mix, xor, perm, mix, xor, perm, mix, xor, perm, mix, xor, perm, mix, xor`, which is six consecutive rounds of "`xor, perm, mix`" followed by an "`xor`".

Users of this machine typically hope to obtain a specific MBTI. Unfortunately, when Doraemon built this machine, he encapsulated the key k inside the machine and refused to tell Kevin the value of the key. Thus, Kevin turned to you, the clever one. Kevin hopes you can construct some inputs (not exceeding 4096, or the machine may be damaged), test run to obtain the corresponding outputs, and then analyze the value of the key. This way, he can write a program to predict the output for each input, allowing everyone to get their desired MBTI.

Can you accomplish this task?

[1] Bitwise XOR refers to performing the modulo 2 addition operation on each corresponding bit of two binary numbers. For example: $(0011)_2 \oplus (0101)_2 = (0110)_2$.

[2] $x \ll y$ means to logically left shift the binary number x by y bits, discarding the overflow and filling the vacated parts with 0. For example: $(1100)_2 \ll 1 = (1000)_2$, $(1001)_2 \ll 1 = (0010)_2$.

[3] $x \ggg y$ means to circularly right shift the binary number x by y bits. For example: $(101)_2 \ggg 1 = (110)_2$, $(110)_2 \ggg 1 = (011)_2$.

Interaction Protocol

To run the MBTI converter, please output "`?` $in_7in_6in_5in_4in_3in_2in_1in_0$" ($in_i \in \{0-9, a-f\}$), where in_i is the hexadecimal value corresponding to the i-th input MBTI. You may perform this operation at most 4096 times. After flushing your output, please read the output of the MBTI converter in the format "$out_7out_6out_5out_4out_3out_2out_1out_0$" ($out_i \in \{0-9, a-f\}$).

To answer for the value of k, please output "`!` k" ($0 \le k < 2^{32}$), where k is the **decimal** representation of the value you believe the key k to be. You may perform this operation at most once. After performing the operation, you should exit your program immediately.

If you run the MBTI converter more than 4096 times, or if your output contains invalid characters, the interactor will output "`-1`". Upon receiving such a response, you should immediately exit your program, and you will receive a "`Wrong Answer`" result. Otherwise, the result is uncertain, as you will attempt to read data from a closed output stream.

To flush your output, you can use:

- `fflush(stdout)` (if you use `printf`) or `cout.flush()` (if you use `cout`) in C and C++.
- `System.out.flush()` in Java and Kotlin.
- `stdout.flush()` in Python.

Example

standard input	standard output
acf7e10b 105092e0	? 04f37255 ? 9090cfca ! 998244353

Note

The following information might be helpful when solving the problem:

1. There are no spaces between the in_is, out_is, so please do not output extra spaces. The output is **not** a hexadecimal number, so **do not** ignore leading 0s that may exist.
2. The range of k is $[0, 2^{32} - 1]$, so please be careful to avoid outputting negative numbers when outputting the answer.
3. The attachment for this problem includes "interact.hpp", which you may need for local debugging. For its usage, you can refer to the file "local_test_example.cpp" provided in the attachment.
4. The interactor is **not adaptive**. That is, the value of k in the interactor is fixed before the interaction begins and will not change according to your interactions.
5. There are 64 tests for this problem in total.