Problem ID: safe 

Korea, a land known for weird but wondrous gadgets. Who would have thought that the hotel you and your friends are staying at has a special safe in each room? Not only can one set the combination to a number that has up to 16 digits, the safe also provides a small hint system in case of weak memory: If one puts in a 3-digit query q, the safe internally divides the combination by q and displays the remainder of the division. How neat is that?

Apparently the previous guest in your room didn't reset their combination and the hotel staff hasn't noticed yet. This gives you some time to think about the safe system. Upon further inspection you notice that some keys on the keypad are broken, so these can be used neither for a query nor for the combination. Also, the number of times one can input a query is limited to 10 and the number of attempts to input the combination is limited to 1. This is probably to prevent abuse.

You come to the conclusion that the safe is in fact not safe, and to prove this point to the others you decide to crack the safe in your room.

Interaction Protocol

Your submission will be interacting with a special program called the *grader*. This means that the output of your submission is sent to the grader and the output of the grader is sent to the standard input of your submission. This interaction must follow a specific protocol:

First, the grader sends a description of the safe in the following format:

- One line containing two integers n, d ($4 \leq n \leq 16, 1 \leq d \leq 3$), the length of the secret combination and the number of broken keys on the keypad respectively.
- One line containing d distinct digits, the digits on the broken keys.

Then, your submission must send requests of the following two types:

- One line of the form "`? q`", where q is a three-digit number. q is allowed to have leading zeroes, but it must not be `000`. The grader will respond with a number r ($0 \leq r < q$), the remainder of dividing the secret combination by q.
- One line of the form "`! c`", where c is an n-digit number (padded with zeroes if necessary), the secret combination determined by your submission.

The numbers q and c must not contain any of the digits with broken keys (this includes leading zeroes in case the `0`-key is broken). It is guaranteed that the secret combination does not contain any of these digits.

After every request you should *flush* the standard output to ensure that the request is sent to the grader. For example, you can use `fflush(stdout)` in C++, `System.out.flush()` in Java, `sys.stdout.flush()` in Python, and `hFlush stdout` in Haskell.

Your submission may send at most 10 requests of the first type.

Your submission must send exactly one request of the second type. After sending this request, it must terminate with exit code `0` as usual.

Your submission will be accepted if it follows the protocol above and it guesses the secret combination correctly. If it sends any invalid request or guesses the combination incorrectly, it will be judged as "Wrong Answer".

Sample Input 1

```
4 3
5 6 9

537

35

107

1
```

Sample Output 1

```


? 800

? 042

? 123

? 008

! 1337
```

In the example interaction above, the output of the grader is on the left and one possible accepted output of a submission is on the right.

The empty lines on both sides only serve to emphasize the chronological order of requests and answers. The grader will never output any empty lines.

Sample Input 2

```
4 3
5 6 9
537
35
107
1
```

Sample Output 2

```
? 800
? 042
? 123
? 008
! 1337
```