Rotation Collation

This is an interactive problem. This means that the input to your program will change based on what your program outputs.

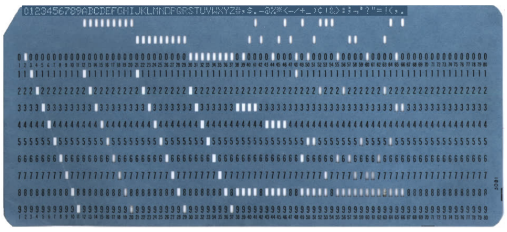


Figure 1: 80-column punch card (1964), Wikipedia

As one of the top contenders in the 1967 Rocky Mountain Regional, you are elated with your team's current progress.

As you finish your turn on the keypunch, you get up to switch seats with one of your teammates and bump the table, spilling your newly created deck onto the floor! Some of the mispunches and other decks from completed problems also spill on the floor. You quickly gather what you think are the cards for your deck and count to make sure that you have the right number.

You hope that you grabbed the right cards and that your deck is still mostly intact. Specifically, you want to know if the deck you have is a rotation of the intended deck, as then you can cut the deck to easily put it back in order.

Your third teammate offers to help you compare the deck with your notes of what the program should be. Each punched card should correspond to a statement in your program. Some statements (like incrementing a variable) are common and may occur multiple times in your deck and program. Given an index into the deck and a statement index in your notes, you and your teammate can quickly compare them to see if that card matches that specific statement.

Please find a valid rotation amount to shift the deck to match the intended program, or indicate that it is not possible.

Interaction

Start by reading an integer N, the size of the deck and number of statements in your program. You are given that $1 \leq N \leq 100\,000$.

You may then make at most $5N$ queries of the form `? i j` (i and j are integers, $1 \leq i, j \leq N$) to check if the i-th card matches the j-th statement in your code. You will receive a wrong answer verdict if you use too many queries.

End each query with a newline character before flushing your output.

```
fflush(stdout); // C
cout.flush(); // C++
System.out.flush(); // Java
print(..., flush=True) # Python
```

The interactor will respond with a `1` if the card and statement match, and a `0` if they do not. The interactor is not adaptive, meaning that, for each test case, the result of each query is known in advance and does not change based on your output.

You must then give your answer in the form `! x`. If there is no valid rotation to make the deck and program match, `x` should be `-1`. Otherwise, `x` should be an integer in $[0, N)$, indicating that each $(i + x)$-th card matches the i-th program statement (wrapping around once you reach the end of the deck).

Your program should then exit with a status code of 0.

If at any point, the interactor receives unexpected input, it will output `-1` and exit. Your program should then also exit cleanly to ensure that you receive a wrong answer verdict.

You are provided with a command-line tool for local testing. The tool has comments at the top to explain its use.

If the state of the deck and program were represented as lists of integers, the sample input would correspond to `1 2 3` for the deck and `3 1 2` for the program, and the deck should be rotated left twice to match the intended program.

Sample Interaction

Lines beginning with $<$ are sent by the judge; lines beginning with $>$ are sent by the program.

```
<3
>? 1 1
<0
>? 2 1
<0
>? 3 1
<1
>? 1 2
<1
>? 2 3
<1
>! 2
```