A Among Us

1s, 256MB **(Interactive Problem)**

Aboard the spaceship *Skeld* are you — the commander — along with N crewmates. The voyage from Earth to Polus was meant to be peaceful, until it wasn't. A series of strange incidents occurs: oxygen runs low, the reactor nearly melts down, the lights flicker, and the doors jam shut. Suspicion spreads — there might be impostors among us.

Moments of crisis were handled with the crewmates' unity, but the tension lingered. Each crewmate now secretly suspects exactly **one** crewmate of being the impostor. Remarkably, **no crewmate is suspected by more than one person**. Formally, there exists a permutation P of length N, where $P[i]$ is the crewmate whom the ***i***-th crewmate suspects.

Overseeing this chaos from the commander's room, you, however, are not actually concerned with finding the impostors. Rather**, your only goal is to determine the hidden permutation** P**.**

Naturally, to achieve this, you may call **emergency meetings**. In each meeting, you can choose the order in which the crewmates speak.

Given a crewmate speaking, they will accuse the crewmate they suspect. If the speaker has not yet been marked as ***sus***, their accusation is valid, and the accused crewmate becomes ***sus***. However, if the speaker is already ***sus***, their accusation is ignored.

Unfortunately, as the commander, you cannot attend the meetings yourself. Instead, after each meeting, the ship's log reports the ***sus*** status of all crewmates as a binary array, where 1 indicates ***sus*** and 0 indicates ***not sus***.

Since time is limited, you may hold at most 400 emergency meetings. Your task is to determine the hidden permutation P within this limit.

INTERACTION

Your program first reads a single integer N ($2 \le N \le 100$) — the number of crewmates.

Then, you may repeatedly perform queries (call emergency meetings). To make a query, print a line in the following format:

- ? q_1 q_2 ... q_N

Here, q_1 q_2 ... q_N is the order in which the crewmates will speak during the meeting. The sequence must be a permutation of length N.

After printing a query, flush the output and read a line containing N integers: s_1 s_2 ... s_N

Here, s_i indicates the status of the i-th crewmate after the meeting ends:

- $s_i = 0$ — the i-th crewmate is not ***sus***.
- $s_i = 1$ — the i-th crewmate is ***sus***.

Each emergency meeting is independent; all crewmates start as not sus.

When you have determined the hidden permutation P, print it in the following format:

- ! p_1 p_2 ... p_N

Printing the answer does **not** count as a query. After printing the final answer, your program should immediately terminate. Failure to do so may result in an undefined verdict.

You may perform at most **400 queries**. If your program performs more than 400 queries or outputs an invalid format, you may receive a Wrong Answer verdict.

The interactor is **not adaptive**, which means that the hidden permutation does not depend on the queries you make.

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

- `fflush(stdout)` or `cout.flush()` in C/C++;
- `System.out.flush()` in Java and Kotlin;
- `sys.stdout.flush()` in Python;

EXAMPLE

INPUT	OUTPUT
6 1 0 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 1 0 1 1 1 0 1	 ? 1 2 3 4 5 6 ? 2 4 3 1 5 6 ? 6 5 4 3 2 1 ? 3 5 4 2 1 6 ! 4 5 3 2 6 1

NOTES

For the first emergency meeting, the permutation P is [4,5,3,2,6,1] and the accusations go as follows:

- Crewmate 1 accuses crewmate 4. Crewmate 4 becomes *sus*.
- Crewmate 2 accuses crewmate 5. Crewmate 5 becomes *sus*.
- Crewmate 3 accuses crewmate 3. Crewmate 3 becomes *sus*.
- Crewmate 4 accuses crewmate 2, but they are already *sus*, so it is ignored.
- Crewmate 5 accuses crewmate 6, but they are already *sus*, so it is ignored.
- Crewmate 6 accuses crewmate 1. Crewmate 1 becomes *sus*.
- The ship reports the status of the crewmates as 1 0 1 1 1 0.