Problem A. Around the Table

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	256 mebibytes

There will be n people numbered from 1 to n, and a room with a round table that has exactly n seats, with each person having a distinct reserved seat that hasn't been revealed yet.

To figure out the reserved seat of each person, up to 60 meetings of the n guests can be organized in the dining room. Before each meeting, you can decide the order in which the guests will arrive. After each meeting, you will receive a list of the guests who arrived earlier than both of their neighbors in the secret seating arrangement at the table.

For each seat, find the person it is reserved for using the information gathered by the result of the meetings.

Interaction Protocol

Before the interaction starts, the judging program reads a permutation of n guests from the test and uses it to answer the queries.

The interaction begins by reading a line with the integer n ($1 \le n \le 10^5$).

Then you can make up to 60 queries.

To make a query, output a line in the format "`?` p_1 p_2 ... p_n" (without quotes) where $p_1, p_2, \ldots, p_n$ is a permutation of length n, the order in which the guests arrive at the meeting. After each query, read an integer k and then k distinct integers $a_1, a_2, \ldots, a_k$ on the same line ($1 \le a_i \le n$), the guests who arrived before their neighbors.

To report the answer, output a line in the format "`!` p_1 p_2 ... p_n" (without quotes), where $p_1, p_2, \ldots, p_n$ is a permutation of length n, the order in which the guests are seated at the round table. You can print any order in which every guest has the same neighbors as the solution.

If a query does not have the format described above, the response you receive will be `-1`. After 60 queries have been made, the response to any other query will be `-1`. Once you receive such a response, terminate the program to receive the "`Wrong Answer`" verdict.

After printing each line, do not forget to output the end of line and flush the output buffer. Otherwise, you will receive the "`Idleness limit exceeded`" verdict. To flush, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python.

Example

standard input	*standard output*
6 2 1 2 2 5 6 2 3 5 2 3 2	 ? 1 2 3 4 5 6 ? 6 5 4 3 2 1 ? 3 4 5 1 6 2 ? 3 1 2 4 5 6 ! 3 4 6 2 5 1