Another Contest 7 Problem 2 - Code Review

Time limit: 0.25s **Memory limit:** 256M

Ren has recently created a website, Ternary Search, which he intends on turning into the premier platform for competitive programmers to talk about anything.

Ren has recently shipped a new feature called code review, where competitive programmers will review the code of other competitive programmers. Ren hopes that this will foster a sense of community as well as help people learn.

Here is how the code review process works: N people each submit their code for review. The ith submission is tagged with two pieces of information - the ID of the user and a timestamp. Both the ID of the user and the timestamp are guaranteed to be a positive integer between 1 and N. Furthermore, user IDs are distinct and timestamps are distinct.

After everyone submits their code, everyone will be assigned a unique timestamp and they will review the code with that timestamp.

There is an obvious flaw in this method though - namely, someone who is unlucky may be assigned to review their own code!

Your job is to find an assignment of submissions to people such that no one reviews their own code.

Constraints

$2 \le N \le 5 \cdot 10^3$

Input/Output Specification

This is an interactive problem.

You will start by reading in a single positive integer N, the number of people who are participating in the code review process.

At this point in time, all N people have submitted their code for review, but you don't know in what order they submitted their code.

Note that per the above, the interactor is **not adaptive**. **As a consequence, the number of test cases for this problem is higher than normal to avoid people hard-coding for the given orderings. The reference solution does not depend on the specific orderings provided in the given test data.**

You can print a tentative assignment of assignments to people by printing a line containing a space-separated permutation of the first N positive integers. The ith integer in the permutation, a_i, indicates that person a_i will be reviewing the code that has timestamp i.

After that, read in a single integer as input. If you read in a zero, your assignment is valid and your program should terminate immediately. Otherwise, you will read in a single positive integer T, indicating that the person who is reviewing the submission with timestamp T is actually reviewing their own code.

The interactor makes no guarantees about which integer T is returned if multiple such integers can be returned. The interactor makes an effort to avoid possible side-channel attacks on how many integers are valid or the location of the first such integer. The interactor will also consume at most 0.025s of CPU time.

If you read in a positive integer, you must propose another assignment. You are allowed to propose up to 20 tentative assignments. If you have proposed 20 assignments and all of them are invalid, the interactor will print `-1` on the final assignment and stop processing further input.

Remember to flush your output before attempting to read in the feedback from the interactor, otherwise the interactor may not return any feedback to you.

If at any point, you output something invalid, the interactor will print `-1` and stop processing further input.

Sample Interaction 1

Note that `<<<` comes before any line that you read in, and `>>>` comes before any line that you output. For this sample, person i's submission was at timestamp i.

```
<<< 3
>>> 1 2 3
<<< 1
>>> 2 1 3
<<< 3
>>> 2 3 1
<<< 0
```

Sample Interaction 2

Note that `<<<` comes before any line that you read in, and `>>>` comes before any line that you output. For this sample, person 1's submission has timestamp 1, person 3's submission has timestamp 2, and person 2's submission has timestamp 3.

```
<<< 3
>>> 1 2 3
<<< 1
>>> 1 3 2
<<< 1
>>> 1 3 2
<<< 2
>>> 1 3 2
<<< 3
>>> 2 3 1
<<< 2
>>> 3 1 2
<<< 3
>>> 3 2 1
<<< 0
```