Task: Zagonetka

Mislav and Marin learned about permutations in their combinatorics class, and they invented an interesting game where the player must guess certain permutations that meet certain conditions. *A permutation* of order n is an array of numbers $p = (p_1, p_2, \ldots, p_n)$ where each number between 1 and n appears exactly once. *A condition* is a pair of distinct numbers (a, b), both between 1 and n, inclusive. A permutation p *meets* condition (a, b) if $p_a < p_b$.

The game is played as follows. Marin first chooses zero or more conditions and one permutation p of order n that meets all of them. In the beginning of the game, Marin texts Mislav only the chosen permutation p (the conditions remain secret). Mislav's goal is to determine the lexicographically smallest and lexicographically largest permutation that meets all of Marin's conditions. In each step of the game, Mislav chooses one permutation q of order n and texts it to Marin. Marin then reveals if that permutation q met all of his secret conditions.

This is an interactive task. Write a program that will play the game instead of Mislav. Your program must, for a given permutation p (of length at most 100) that meets the secret conditions, in at most 5 000 steps find the lexicographically smallest and the lexicographically largest permutation that meet all the conditions.

Interaction

Before the interaction, your program must read from the standard input the following: The first line of input contains the integer n — the size of all permutations in the game. The following line contains n distinct integers $p_1, p_2, \ldots, p_n$ $(1 \leq p_j \leq n)$ — the permutation p. You can assume that p meets all of Marin's conditions.

After this, your program can send Marin *queries* by writing to the standard output. Each query must be printed in its own line in the form of "`query` $q_1\ q_2\ \ldots\ q_n$" where $q_1, q_2, \ldots q_n$ are distinct integers between 1 and n, inclusively. After each printed query, your program must *flush* the output and read from the standard input Marin's *reply* — the number 1 if the permutation q from the query meets all the conditions, and 0 otherwise.

When your program has found a solution, it must output a line to the standard output containing the command "`end`", then a line of the form of "$a_1\ a_2\ \ldots\ a_n$" containing the required lexicographically smallest permutation and a line of the form of "$b_1\ b_2\ \ldots\ b_n$" containing the required lexicographically largest permutation. In the end, the program must *flush* the output again and terminate the execution.

Please note: Using the evaluation system, you can obtain code samples that perform a valid interaction, including the *flush* of the output.

Interaction sample

In the following interaction sample, the first column contains the data that your program outputs to the standard output, and the second column contains the data that your program reads from the standard input. After three steps of the game, i.e. after three queries, the program determines the correct solution.

Output	Input	Please note
	`4`	the chosen secret conditions are $(2, 1)$ i $(3, 4)$
	`3 2 1 4`	
`query 2 3 1 4`	`0`	condition $(2, 1)$ is not met
`query 3 2 4 1`	`0`	condition $(3, 4)$ is not met
`query 4 1 2 3`	`1`	both conditions are met
`end`		
`2 1 3 4`		
`4 3 1 2`		

Testing

You can test your solutions in two ways, locally or using the evaluation system. For both variants, you must first create an input file that contains the test case for which you want to test your program. The format of the input file must be according to the following rules: The first line of input contains the integer n — the size of all the permutations in the game. The following line contains n different integers $p_1, p_2, \ldots, p_n$ $(1 \leq p_j \leq n)$ — permutation p. The following line contains the integer m $(0 \leq m \leq 10\,000)$ — the number of conditions. Each of the following m lines contains two distinct integers a and b $(1 \leq a, b \leq n)$ that represent a single condition. Permutation p must meet all m conditions.

For example, an input that corresponds to the above interaction sample is:

```
4
3 2 1 4
2
2 1
3 4
```

In order to test using the evaluation system, you first need to upload the source code of your solution using the page "Submit", and then send the test case using the page "Test". Local testing is done using the file "`zagonetka_test`" which is possible to download using the evaluation system. Testing is performed with the following command:

```
./zagonetka_test ./your_solution input_file
```

When testing using the evaluation system, as output you will get the information whether your program solved your test case correctly, and the information about the commands your program ordered, and the responses it got.

When testing locally, you will not get the information whether your program solved the test case correctly. You will need to make sure of that yourselves. The information about the course of the game will be written to the file `zagonetka.log` in the current folder.

Scoring

Subtask	Score	Constraints
1	9	$2 \leq n \leq 6$
2	18	$30 \leq n \leq 70$, Marin chose exactly 1 condition
3	22	$10 \leq n < 30$
4	51	$70 < n \leq 100$