Code Jam 2022 - Round 3

Revenge of GoroSort

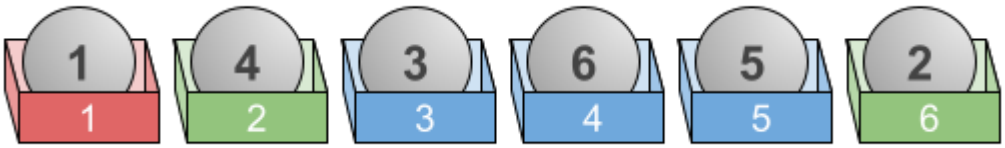
Problem

In this problem, when something is said to be chosen at random, it means uniformly at random from among all valid possibilities, and independently of any other choice.

Code Jam contestants once helped the mighty Goro sort an array of integers. (You do not need to read that problem to solve this one.) Once again, Goro needs your help. He has $\mathbf{N}$ boxes lined up on the table in a single row, numbered 1 through $\mathbf{N}$ from left to right. Each box has exactly one ball inside. Balls are also numbered 1 through $\mathbf{N}$. Goro wants ball i to end up in box i, for all i. That is, he wants to leave the balls in sorted order. Unfortunately, that is not initially the case.

When Goro bumps the table with his powerful fists, balls pop up in the air and fall back in boxes. Goro can do this so accurately that exactly one ball falls into each box. A ball may fall into the same box it came out of, or into a different one.

Better yet, Goro also has the ability to assign colors to boxes before each bump. Then, he can bump the table in such a way that balls coming out of a box of color c always fall into a box of color c. As impressive as this accuracy is, Goro does not have any more control than that. Within each color, balls end up assigned to boxes at random.



For example, suppose the balls appear in the order $1, 4, 3, 6, 5, 2$ (as seen above). He might choose — not necessarily optimally — to give the first box the color red, the second and sixth boxes the color green, and the third through fifth boxes the color blue. Then, after Goro bumps the table,

- The 1 in the first box falls back into the same box, because that is the only red box.
- The 4 and 2 in the second and sixth boxes remain in place with probability $\frac{1}{2}$, and switch places with probability $\frac{1}{2}$.
- The $3, 6, 5$ in the third, fourth, and fifth boxes end up in one of the following orders, each with probability $\frac{1}{6}$:
  - $3, 6, 5$
  - $3, 5, 6$
  - $6, 3, 5$
  - $6, 5, 3$
  - $5, 3, 6$
  - $5, 6, 3$

So, for example, the probability of the bump leaving the balls in the order $1, 2, 3, 5, 6, 4$ is $\frac{1}{12}$. If Goro got this or some other non-sorted result, he would have to designate a set of box colors for the next round, and so on, until he eventually arrives at the sorted $1, 2, 3, 4, 5, 6$. Goro can assign colors to boxes in any way before each bump, regardless of previous assignments.

Can you help Goro implement a better strategy that will efficiently sort the balls? It is guaranteed that the balls start in a random non-sorted order.

Input and output

This is an interactive problem. You should make sure you have read the information in the Interactive Problems section of our FAQ.

Initially, your program should read a single line containing three integers, $\mathbf{T}$ $\mathbf{N}$ $\mathbf{K}$: the number of test cases, the number of boxes per test case, and the total number of bumps allowed for all test cases combined. Then, $\mathbf{T}$ test cases must be processed.

Each test case begins with the judge sending one line with $\mathbf{N}$ integers, with each integer from 1 to $\mathbf{N}$ appearing exactly once, and with the list chosen at random from all non-sorted lists. Then you must engage in a series of interactions with the judge. Each interaction works as follows:

- You send one line of $\mathbf{N}$ integers $C_1, C_2, \ldots, C_{\mathbf{N}}$, in which each integer is between 1 and $\mathbf{N}$, inclusive. Each C_i represents that you are assigning color C_i to box i for the next bump. You may choose how many colors there are and how they are numbered, but you must assign a color to each box.
- The judge simulates the bump as explained in the statement. If this results in the balls being in sorted order:
  - If this interaction was the $\mathbf{K}$-th interaction across all test cases, and this was not the last test case, the judge sends one line with the integer -1 and does not output anything further.
  - Otherwise, the judge sends one line with the integer 1 and then immediately begins the next test case, if there is one. If this was the last test case, your program must exit without error and without sending anything further.

- Otherwise, the balls are not sorted, and:
  - If this interaction was the $\mathbf{K}$-th across all test cases, or if you provided an invalid line (e.g., too few integers, or color numbers out of range), the judge sends one line with the integer -1 and does not output anything further.
  - If this was not your $\mathbf{K}$-th interaction, the judge sends one line with the integer 0, and then another line with $\mathbf{N}$ integers, with each integer from 1 to $\mathbf{N}$ appearing exactly once, and in non-sorted order, representing the new order of the balls, that is, the i-th of these integers is the ball that fell into box i. Then you must begin another interaction.

As usual, if the memory limit is exceeded, or your program gets a runtime error, you will receive the appropriate judgment. Also, if your program continues to wait for the judge after receiving a -1, your program will time out, resulting in a Time Limit Exceeded error. Notice that it is your responsibility to have your program exit in time to receive a Wrong Answer judgment instead of a Time Limit Exceeded error.

Be advised that the judge uses the same source of randomness each time, so in the absence of other errors (e.g. Time Limit Exceeded, Memory Limit Exceeded), submitting the exact same code twice will yield the same outcome twice.

Limits

Time limit: 20 seconds.
Memory limit: 1 GB.
$\mathbf{T} = 1000$.
$\mathbf{N} = 100$.

Test Set 1 (Visible Verdict)

$\mathbf{K} = 16500$.

Test Set 2 (Visible Verdict)

$\mathbf{K} = 12500$.

Test Set 3 (Visible Verdict)

$\mathbf{K} = 11500$.

Testing Tool

You can use this testing tool to test locally or on our platform. To test locally, you will need to run the tool in parallel with your code; you can use our interactive runner for that. For more information, read the instructions in comments in that file, and also check out the Interactive Problems section of the FAQ.

Instructions for the testing tool are included in comments within the tool. We encourage you to add your own test cases. Please be advised that although the testing tool is intended to simulate the judging system, it is **NOT** the real judging system and might behave differently. If your code passes the testing tool but fails the real judge, please check the Coding section of the FAQ to make sure that you are using the same compiler as us.

Download testing tool

Sample Interaction

Judge **Solution**

Number of cases, number of boxes per case, number of interactions allowed across all cases

```
2 4 8
```

Case 1

```
1 4 3 2
```

Judge gives this non-sorted list.

```
1 2 3 2
```

Solution assigns color 1 to the first box, color 2 to the second and fourth boxes, and color 3 to the third box.
The judge trivially assigns the balls in the boxes of color 1 and 3 back to their original box, and randomly chooses whether to swap the balls in the second and fourth boxes. In this case, it does not swap them.

```
0
1 4 3 2
```

```
1 2 3 2
```

Solution does the same thing as before.
This time the judge does happen to swap the 4 and the 2.

```
1
```

Judge indicates that this test case was solved, and immediately begins the next case.

Case 2

```
2 1 4 3
```

```
4 4 4 4
```

Solution assigns all boxes color 4. Notice that it's OK to not have colors 1, 2, or 3.
The judge chooses a new random order for the balls. Wow, what luck, they come out in order!

```
1
```

Judge indicates that this test case was solved, and sends no further output, because this was the last test case.
Contestant solution should exit (with no error) to avoid a Time Limit Exceeded error.

Note that the sample interaction does not satisfy the constraints of any of the test sets. It is only presented to clarify the input and output format.