2080. Wallet

Time limit: 2.0 second
Memory limit: 256 MB

Eugene is not a student struggling to pay his bills anymore. Now he lives in a big city and works for a well-known IT-company. He forgot the times when his wallet contained only his father's credit card with no money on it; now it's full with Eugene's own various credit, discount and membership cards.

Every week Eugene follows the same routine: he visits the same places in the same order and talks with the same people. There is no place for surprises and complications in his life. Finally, after several months in a strange city, he can just perform a pre-determined order of actions and not trouble himself with anything.

However, Eugene does have one problem: sometimes in a store, restaurant or even at a subway station entrance he has to search his wallet for the right card for more than 5 seconds. Eugene hates to waste his precious time on that.

Cards in Eugene's wallet are stored as a stack. Ha wants to be able to just take the top one every time he needs to pay or get a discount for something. After using a card he has no problem with inserting it in any position of the stack.

Knowing an order in which cards are going to be used throughout the week one can easily find a way to assemble them so that the top one will always be the right card to use. Eugene could have done that himself, but he doesn't feel like solving this problem. You do it.

Input

The first line of the input contains two integers n and k ($1 \le n, k \le 10^5$) — total number of cards Eugene has in his wallet and the number of times he uses them during the week. The cards are enumerated with integer numbers from 1 to n.

The second line contains k numbers $a_1, a_2, \ldots, a_k$ ($1 \le a_i \le n$), separated by spaces: cards' numbers in the order they are to be used.

Output

Output k+1 lines.

The first line should contain n card numbers separated by spaces from 1 to n — the initial order of cards in the stack, from top to bottom.

The line number (i + 1) should contain one number — how many cards will be located higher in the stack than the card a_i after it has been used and returned into the wallet.

If there are several correct solutions, output any of them.

Sample

input	output
3 5 3 1 2 2 1	3 1 2 2 1 0