A. Sorting by Subsequences

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a sequence $a_1, a_2, ..., a_n$ consisting of **different** integers. It is required to split this sequence into the **maximum** number of subsequences such that after sorting integers in each of them in increasing order, the total sequence also will be sorted in increasing order.

Sorting integers in a subsequence is a process such that the numbers included in a subsequence are ordered in increasing order, and the numbers which are not included in a subsequence don't change their places.

Every element of the sequence must appear in exactly one subsequence.

Input

The first line of input data contains integer n ($1 \le n \le 10^5$) — the length of the sequence.

The second line of input data contains n different integers $a_1, a_2, ..., a_n$ ($-10^9 \le a_i \le 10^9$) — the elements of the sequence. It is guaranteed that all elements of the sequence are distinct.

Output

In the first line print the maximum number of subsequences k, which the original sequence can be split into while fulfilling the requirements.

In the next k lines print the description of subsequences in the following format: the number of elements in subsequence c_i ($0 < c_i \le n$), then c_i integers $l_1, l_2, ..., l_{c_i}$ ($1 \le l_j \le n$) — indices of these elements in the original sequence.

Indices could be printed in any order. Every index from 1 to n must appear in output **exactly once**.

If there are several possible answers, print any of them.

Examples

input
6 3 2 1 6 5 4

output
4 2 1 3 1 2 2 4 6 1 5

input
6 83 -75 -49 11 37 62

output
1 6 1 2 3 4 5 6

Note

In the first sample output:

After sorting the first subsequence we will get sequence $1\ 2\ 3\ 6\ 5\ 4$.

Sorting the second subsequence changes nothing.

After sorting the third subsequence we will get sequence $1\ 2\ 3\ 4\ 5\ 6$.

Sorting the last subsequence changes nothing.