Permutation Compression II

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

For an element in an array, if it is **non-zero** and it is **strictly greater** than all of the elements before it, then it is considered as a beautiful element. We define the beauty of an array as the number of beautiful elements in the array.

There is a permutation of length n. You want to maximize its beauty by changing some of the **beautiful** elements into zero. Note that some elements may become beautiful after your operation, and it will become available as a target of your operation.

Since modifying the array is tiring, you decide to maximize the beauty of the array by using the minimum number of operations possible.

If there are multiple solutions, output any.

Input

There are multiple test cases.

The first line contains an integer t ($1 \leq t \leq 10^6$), denoting the number of test cases. For each test case:

The first line contains an integer n ($1 \leq n \leq 10^6$), denoting the size of the permutation.

The next line contains n integers p_i ($1 \leq p_i \leq n$), denoting the elements in the permutation. It is guaranteed that all p_i in one test case are distinct.

It is guaranteed that $\sum n \leq 10^6$.

Output

For each test case:

The first line contains two integers b, c, denoting the maximum beauty of the modified array and the minimum number of operations.

The second line contains c integers o_i, denoting the operation sequence. Operation o_i means to change the o_i-th element into zero. You should make sure that the o_i-th element is beautiful before your i-th operation.

If there are multiple solutions, output any.

Example

standard input	standard output
2 3 3 1 2 3 1 2 3	2 1 1 3 0