B1. Social Network (easy version)

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The only difference between easy and hard versions are constraints on n and k.

You are messaging in one of the popular social networks via your smartphone. Your smartphone can show at most k most recent conversations with your friends. Initially, the screen is empty (i.e. the number of displayed conversations equals 0).

Each conversation is between you and some of your friends. There is at most one conversation with any of your friends. So each conversation is uniquely defined by your friend.

You (suddenly!) have the ability to see the future. You know that during the day you will receive n messages, the i-th message will be received from the friend with ID id_i ($1 \le id_i \le 10^9$).

If you receive a message from id_i in the conversation which is currently displayed on the smartphone then nothing happens: the conversations of the screen do not change and do not change their order, you read the message and continue waiting for new messages.

Otherwise (i.e. if there is no conversation with id_i on the screen):

- Firstly, if the number of conversations displayed on the screen is k, the last conversation (which has the position k) is removed from the screen.
- Now the number of conversations on the screen is guaranteed to be less than k and the conversation with the friend id_i is not displayed on the screen.
- The conversation with the friend id_i appears on the first (the topmost) position on the screen and all the other displayed conversations are shifted one position down.

Your task is to find the list of conversations (in the order they are displayed on the screen) after processing all n messages.

Input

The first line of the input contains two integers n and k ($1 \le n, k \le 200$) — the number of messages and the number of conversations your smartphone can show.

The second line of the input contains n integers $id_1, id_2, \ldots, id_n$ ($1 \le id_i \le 10^9$), where id_i is the ID of the friend which sends you the i-th message.

Output

In the first line of the output print one integer m ($1 \le m \le min(n, k)$) — the number of conversations shown after receiving all n messages.

In the second line print m integers $ids_1, ids_2, \ldots, ids_m$, where ids_i should be equal to the ID of the friend corresponding to the conversation displayed on the position i after receiving all n messages.

Examples

input Copy

```
7 2
1 2 3 2 1 3 2
```

output Copy

```
2
2 1
```

input Copy

```
10 4
2 3 3 1 1 2 1 2 3 3
```

output Copy

```
3
1 3 2
```

Note

In the first example the list of conversations will change in the following way (in order from the first to last message):

- $[]$;
- $[1]$;
- $[2, 1]$;
- $[3, 2]$;
- $[3, 2]$;
- $[1, 3]$;
- $[1, 3]$;
- $[2, 1]$.

In the second example the list of conversations will change in the following way:

- $[]$;
- $[2]$;
- $[3, 2]$;
- $[3, 2]$;
- $[1, 3, 2]$;
- and then the list will not change till the end.