Task	Test Cases	Points/case	Total Points	Memory Limit	Time Limit
Keys	16	A: 1 B: 5	100[1]	16 MB	1s
Samba	20	5	100	2 MB	1s
Lego	10	10	100	64 MB	1s

[1] Any contestant to gets all cases correct (i.e. 96 points) will receive 4 bonus points.

Task 1: Keys

Consider a table with rows and columns. The columns are numbered from 1 to C. For simplicity's sake, the items in the table are strings consisting of lower case letters.

Col. 1	Col. 2	Col. 3
apple	red	sweet
apple	green	sour
pear	green	sweet
banana	yellow	sweet
banana	brown	rotten

Table 1

Col. 1	Col. 2	Col. 3
banana	brown	rotten
apple	green	sour
pear	green	sweet
apple	red	sweet
banana	yellow	sweet

Table 2

Col. 1	Col. 2	Col. 3
apple	green	sour
apple	red	sweet
banana	brown	rotten
banana	yellow	sweet
pear	green	sweet

Table 3

You are given the operation Sort(k) on such tables: Sort(k) sorts the rows of a table in the order of the values in column k (while the order of the columns does not change). The sort is stable, that is, rows that have equal values in column k, remain in their original order. For example, applying Sort(2) to Table 1 yields Table 2.

We are interested in sequences of such sort operations. These operations are successively applied to the same table. For example, applying the sequence Sort(2); Sort(1) to Table 1 yields Table 3.

Two sequences of sort operations are called equivalent if, for any table, they have the same effect. For example, Sort(2); Sort(2); Sort(1) is equivalent to Sort(2); Sort(1). However, it is not equivalent to Sort(1); Sort(2), because the effect on Table 1 is different.

Your task is, given a sequence of sort operations, determine a shortest equivalent sequence.

Input

The first line of input contains two integers, **C** and **N**. **C** ($1 \le$ **C** $\le 1\,000\,000$) is the number of columns and **N** ($1 \le$ **N** $\le 3\,000\,000$) is the number of sort operations.
The second line contains **N** integers, $\mathbf{k_i}$ ($1 \le \mathbf{k_i} \le$ **C**).
It defines the sequence of sort operations Sort(k_1); ...; Sort(k_N).

Output

The first line of output contains one integer, **M**, the length of the shortest sequence of sort operations equivalent to the input sequence (Subtask A). The second line contains exactly **M** integers, representing a shortest sequence (Subtask B). You can omit the second line if you solve only Subtask A.

Example

Input	Output
4 6 1 2 1 2 3 3	3 1 2 3

Task 2: Samba

Every year Rio de Janeiro is the host of The Rio samba dance festival. In this year's edition the **n** dancers representing the samba schools around the world will dance on the streets trying to impress the public by both dance and clothes. **Each samba school** is represented by a single group of dancers and has a unique identification number (**ID**) which will be worn by all of its members. Within a group, all members are wearing the same suits, with the identification number attached, when moving **in formation** on the streets of Rio.

For everything to go well, the organizers require each group to arrange its members on multiple rows, each formed exactly of **k** dancers.

Your task is, knowing that **only one samba school** couldn't organize its dancers according to the conditions, to find its ID.

Input

The first line of the input contains the space-separated numbers **n** ($1 \le n \le 1\,000\,000$) and **k** ($2 \le k \le 1000$).

The following **n** lines contain the IDs ($C_1, C_2, \ldots, C_n$) of the **n** dancers, where $\mathbf{0 \le C_i \le 1\,000\,000\,000}$

Output

The first line of the output contains a single value, representing the answer to the task.

Special cases

40% of tests has **n** ≤ 1000

Example

Input	Output

11 2
123
1678
43
123
123
43
123
43
123
1678
123
	43

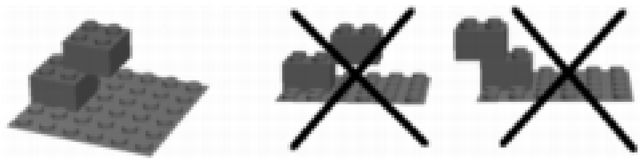
There are **11** dancers which belong to the samba schools identified by the IDs **123**, **1678** and **43**. Dancers of each samba school have to arrange on rows formed of exactly **2** persons. One samba school has the **123** ID and **6** dancers which can be arranged on **3** rows, each consisting of **2** persons. Another samba school, with the **1678** ID, has **2** dancers which can be arranged on a single row.

The school with the ID **43** has **3** dancers which cannot be organized according to the conditions.

Task 3: Lego

You are using Lego building blocks to train an artificial vision system. Write a program that, given pictures of a Lego construction taken from two angles, calculates in how many different ways it can be built.

In this task, there is only one kind of lego block (with 2 × 2 "knobs", see picture below), but it can have three different colors: white (W), gray (G) or black (B). All blocks exist in unlimited amounts. You use a square base with 6 × 6 knobs. Every block must have its edges parallel to this base and no block may extend outside of it. Every block must rest upon at least one underlying block.


Left: An allowed way to place a block on top of another one. Center: An illegal way (the upper block hangs in the air). Right: Another illegal way (the upper block extends outside the base).

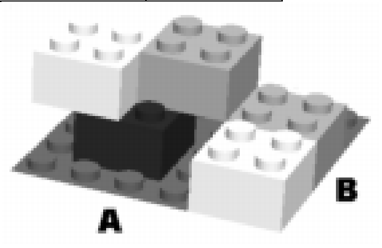
Input.
The first line of input contains **H** (1 ≤ **H** ≤ 6), the height of the construction. Then follow **H** lines with 6 characters on each line, giving a picture of the construction as seen from one side (marked A on the figure below). The *j*th character on the *i*th line specifies what you see looking at the *j*th column from the left on the *i*th row from above. Each character may be one of 'W', 'G', 'B' or '.', specifying a color ('W', 'G', or 'B') or a hole ('.'). Note that you cannot estimate the depth, so a color seen in a certain position may either belong to a block near the front edge, or further back, provided no other block is blocking the sight.
The first picture is followed by another set of **H** lines with the construction seen from an angle where the observer has moved 90 degrees counterclockwise around the construction (marked B on the figure below).

Output.
The program should output one line containing a single integer: the number of different Lego constructions that satisfy the pictures given in the input. Note that even if two different possible constructions could be obtained from each other by rotating or mirroring, they both should be counted. For the given input, the answer will always fit in a signed 64-bit integer.

Example

Input	Output

`WWGG..`
`.BB.WW`
`.WGG..`
`WWGG..`
	`26`



One of the possible constructions in the example.