TASK	**SAHOVNICA**	**POREDAK**	**MALCOLM**	**AERODROM**	**HERKABE**	**PROCESOR**

source code	`sahovnica.pas`
	`sahovnica.c`
	`sahovnica.cpp`
		`poredak.pas`
		`poredak.c`
		`poredak.cpp`
			`malcolm.pas`
			`malcolm.c`
			`malcolm.cpp`
				`aerodrom.pas`
				`aerodrom.c`
				`aerodrom.cpp`
					`herkabe.pas`
					`herkabe.c`
					`herkabe.cpp`
						`procesor.pas`
						`procesor.c`
						`procesor.cpp`

input	standard input (*stdin*)					
output	standard output (*stdout*)					
time limit	1 second	1 second	1 second	1 second	1 second	1 second
memory limit	32 MB	32 MB	32 MB	32 MB	32 MB	32 MB
point value	**50**	**80**	**100**	**120**	**140**	**160**
	650					

Problem authors: **Adrian Satja Kurdija** (1 – 5), **Ivan Katanić** (6)
Problems translated from Croatian by: **Ivan Pilat**

Mirko has become a hardcore patriot, so he has asked you to draw him a Croatian chessboard (checkerboard).

The chessboard consists of red and white cells. The upper left cell is red, with the remaining cells alternating between white and red in rows as well as columns. We will represent red areas with 'X' (**uppercase** letter x) characters, and white areas with '.' (period) characters.

Mirko's chessboard should consist of **R** × **C** cells, that is, **R** rows and **C** columns. Each row should be **A** characters high, and each column **B** characters wide. Consider the sample tests below for further clarification.

INPUT

The first line of input contains two positive integers **R** and **C** (1 ≤ **R**, **C** ≤ 10) from the problem statement.

The second line of input contains two positive integers **A** and **B** (1 ≤ **A**, **B** ≤ 10) from the problem statement.

OUTPUT

The output must consist of a total of **R** * **A** rows and **C** * **B** columns, forming the chessboard described above.

SAMPLE TESTS

input 2 4 2 2 **output** XX..XX.. XX..XX.. ..XX..XX ..XX..XX	**input** 5 5 2 3 **output** XXX...XXX...XXX XXX...XXX...XXX ...XXX...XXX... ...XXX...XXX... XXX...XXX...XXX XXX...XXX...XXX ...XXX...XXX... ...XXX...XXX... XXX...XXX...XXX XXX...XXX...XXX

Mirko-wan has just received the results of his history exam. One of the problems was putting famous historical battles into chronological order. The correct order was:

1. Blockade of Naboo 2. Battle of Geonosis 3. Battle of Yavin 4. Battle of Hoth 5. Battle of Endor

Mirko-wan has studied (relatively) hard for the exam, so he remembered the exact years of all battles, except for the Blockade of Naboo. He couldn't remember anything about it, so he randomly placed it last instead of first, obtaining the order:

1. Battle of Geonosis 2. Battle of Yavin 3. Battle of Hoth 4. Battle of Endor 5. Blockade of Naboo

Since Mirko-wan's order doesn't match the correct solution at any index, Mirko-wan's score on that problem was – to his disappointment – 0 out of 5 points, even though he knew the correct order of four out of five battles!

This opens the question of fair scoring of an ordering problem. The example given above suggests that scoring by counting the number of items in the correct absolute position isn't fair. Is there a better way?

One possibility is finding the longest subsequence (not necessarily contiguous) of correctly ordered items. This isn't the best solution either: if an item is displaced by just one position from the correct order, the score that it awards drops to zero, even though it was almost correctly ordered.

Mirko-wan has thus suggested (using the MAWT – Might As Well Try a Mind Trick approach) the following scoring method to his history teacher. **For every two items**, the student will receive **1 point** if the two items are in mutually correct order. In other words, the number of points is the number of item pairs that the student has correctly ordered. The maximum number of points is then, of course, the total number of pairs, which equals **N** * (**N** - 1) / 2, where **N** is the total number of entries.

INPUT

The first line of input contains the positive integer **N** (2 ≤ **N** ≤ 2500), the number of items. The items are distinct words consisting of 3 to 15 lowercase English letters.

The second line of input contains the **N** items, space-separated, listed in the correct order.

The third line of input contains the **N** items, space-separated, listed in Mirko-wan's order.

OUTPUT

The first and only line of output must contain, with no spaces, the following: the number of points that Mirko-wan would earn using his scoring method, a / (forward slash) character, and the maximum possible number of points for that problem (again assuming Mirko-wan's scoring method). (This is the usual notation found on a typical graded exam.)

SAMPLE TESTS

input	input
3 alpha beta gamma alpha gamma beta	5 naboo geonosis yavin hoth endor geonosis yavin hoth endor naboo
output	**output**
2/3	6/10

Clarification of the first example: Mirko-wan has received points for the pairs (alpha, beta) and (alpha, gamma).

Since teacher Herkabe has started ranking his **N** students, the number of friendships in his class has sharply fallen. The students near the bottom of the rankings list have become jealous of the top students, while the top students started looking down on their less successful colleagues.

According to Malcolm's observations, the following rule holds: two students are **friends** if their ranks are close enough, more precisely, if they differ by **at most K**. For example, if **K** = 1, then only neighbouring students on the rankings list are friends. Furthermore, two students are **good friends** if they are **friends** and their **names have the same length**.

Write a program to calculate the **number of pairs of good friends** in this gifted class.

INPUT

The first line of input contains two positive integers, **N** (3 ≤ **N** ≤ 300 000) and **K** (1 ≤ **K** ≤ **N**), from the problem statement.

Each of the following **N** lines contains a single student's name. The names are given in the order they appear on the rankings list. They consist of between 2 and 20 (inclusive) uppercase English letters.

OUTPUT

The first and only line of output must contain the required number of pairs.

SAMPLE TESTS

<table>
<tr>
<td>

input

```
4 2
IVA
IVO
ANA
TOM
```

output

```
5
```

</td>
<td>

input

```
6 3
CYNTHIA
LLOYD
STEVIE
KEVIN
MALCOLM
DABNEY
```

output

```
2
```

</td>
</tr>
</table>

The Croatian delegation, consisting of **M** people, is travelling to IOI 2013 in Australia[1]. They are currently waiting in a queue for check-in at the airport. There are **N** check-in desks open. Some officials work more efficiently than others, so the desks operate at different speeds. At the **k**-th desk, $\mathbf{T_k}$ seconds are required to finish check-in of a single passenger, and members of our delegation happen to know the exact numbers.

In the beginning, all desks are ready to accept the next passenger, and the delegation members are the only people in the queue. A person can only occupy (start check-in at) an available desk when all people in front of that person in the queue have left the queue (started, not necessarily finished, check-in) already. At that moment, the person can immediately occupy an available desk (if there is one), but can also choose to wait for another (faster) desk to become available. Our delegation members, being computer science geeks, make this decision in such a way that the **moment when all of them have finished check-in is as soon as possible**. Your task is finding that moment in time.

Let us describe the scenario from the first example below. There are two desks, with processing times of 7 and 10 seconds, respectively. Out of the six people in the delegation, the first two immediately occupy the two desks. At time 7, the first desk is freed, and the third person occupies it. At time 10, the fourth person occupies the second desk. At time 14, the fifth person occupies the first desk. At time 20, the second desk is freed again, but the sixth person decides to wait another second (time 21) for the first desk to become available, and then occupy it. This way, the check-in is completed by time 28. If the sixth person hadn't waited for the faster desk, the check-in would have taken a total of 30 seconds.

INPUT

The first line of input contains two positive integers, **N** ($1 \le \mathbf{N} \le 100\ 000$), the number of desks, and **M** ($1 \le \mathbf{M} \le 1\ 000\ 000\ 000$), the number of people in the delegation.

Each of the following **N** lines contains a number $\mathbf{T_k}$ from the problem statement ($1 \le \mathbf{T_k} \le 10^9$).

OUTPUT

The first and only line of output must contain the required minimum time in seconds.

SCORING

In test data worth a total of 75 points, the number **M** will be at most 300 000.

[1] Assuming, of course, that the Apocalypse didn't happen.

SAMPLE TESTS

input	input
2 6 7 10	7 10 3 8 3 6 9 2 4
output	**output**
28	8

Teacher Herkabe has decided to rank his students again. This time, he wants his list to also be aesthetically pleasant, so he has decided that similar names (those beginning with the same letter or sequence of letters) must be close to one another on the list. Therefore, he has devised the following rule:

For every two names on the list that begin with the same letter sequence, all names between them on the list must also begin with that letter sequence.

For example, consider the names MARTHA and MARY (characters from a beautiful story). They both begin with the sequence MAR, so names beginning with the same sequence (like MARCO and MARVIN) can appear in between (but not MAY, for example).

Notice that the lexicographically sorted ordering always satisfies this rule, but it is by no means the only valid ordering. Your task is determining how many different orderings satisfy the rule, i.e. how many options teacher Herkabe has for his ranking list.

INPUT

The first line of input contains the positive integer **N** ($3 \le \mathbf{N} \le 3000$), the number of names.

Each of the following **N** lines contains a single name: a sequence of between 1 and 3000 (inclusive) uppercase English letters. The names are distinct and given in no particular order.

OUTPUT

The first and only line of output must contain the required number of possible ranking lists, modulo 1 000 000 007.

SCORING

In test data worth a total of 60 points, the number **N** will be smaller than 10.

SAMPLE TESTS

input	input	input
3 IVO JASNA JOSIPA	5 MARICA MARTA MATO MARA MARTINA	4 A AA AAA AAAA
output	**output**	**output**
4	24	8

Mirko has received an interesting homework assignment: to design his own little processor (Mirkoprocessor). The processor has **N** registers with indices from 1 to **N**, and each register holds one unsigned 32-bit integer in the usual binary format (the possible values range from 0 to 2^{32} - 1).

The processor is capable of executing the following instruction types:

Instruction type	**Description**	**Example**
1 **K M**	Rotate the bits of register **K** by **M** positions to the right; write the result back to register **K**.	00000000000000000010001111111011 → (**M** = 1010) → 11111110110000000000000000001000 *(in base 10: 9211 → (**M** = 10) → 4 273 995 784)*
2 **K L**	Compute the bitwise XOR of registers **K** and **L**; output the result to the system bus.	00000000000000000000001111000111 XOR 00000000000001111100000000000111 = 00000000000001111100001111000000 *(in base 10: 967 XOR 507 911 = 508 864)*

Mirko has already built a model of the processor, and only then realized that he'd forgotten to include an operation to read the contents of a register. Now, his only option is to execute a large number of type 1 and type 2 instructions and infer the register contents from the results. Having executed a sequence of commands, he has asked you to help him derive the initial register contents consistent with the obtained results.

If there are multiple possibilities for the initial register state combination, find the **lexicographically smallest** one. (If two combinations have equal values in the first **K** – 1 registers and different values in register **K**, the lexicographically smaller combination is the one with the smaller value in register **K**.)

INPUT

The first line of input contains two positive integers: **N** (2 ≤ **N** ≤ 100 000), the number of registers, and **E** (1 ≤ **E** ≤ 100 000), the number of executed instructions.

The remaining input lines describe the instructions in the order that they were executed by Mirkoprocessor, formatted as described in the table above, one per line. All instructions are legal (the following conditions hold: 1 ≤ **K**, **L** ≤ **N**, 0 ≤ **M** < 32). Each instruction of **type 2** is followed by another line containing a positive integer between 0 and 2^{32} – 1, inclusive – the result of that operation (the bitwise XOR value) in base 10.

OUTPUT

The first and only line of output must contain the required **N** register values, separated by spaces.

If there is no possible combination of initial values consistent with input, output only the number -1, to notify Mirko that his processor has a bug.

SCORING

In test data worth a total of 64 points, the numbers **N** and **E** will be smaller than 1000.

SAMPLE TESTS

input	input	input
3 3 2 1 2 1 2 1 3 2 2 2 3 3	4 6 2 4 2 3 2 4 1 6 1 3 1 2 3 1 2 1 2 2 2 2 3 7	5 6 2 4 2 10 2 5 3 2 2 2 3 1 2 1 4 3 1 3 1 2 3 4 2147483663
output	**output**	**output**
0 1 2	5 0 14 3	15 6 7 12 5