Codeforces Beta Round #99 (Div. 1)

A. Literature Lesson

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

Vera adores poems. All the poems Vera knows are divided into quatrains (groups of four lines) and in each quatrain some lines contain rhymes.

Let's consider that all lines in the poems consist of lowercase Latin letters (without spaces). Letters "`a`", "`e`", "`i`", "`o`", "`u`" are considered vowels.

Two lines rhyme if their suffixes that start from the k-th vowels (counting from the end) match. If a line has less than k vowels, then such line can't rhyme with any other line. For example, if $k = 1$, lines $commit$ and $hermit$ rhyme (the corresponding suffixes equal it), and if $k = 2$, they do not rhyme ($ommit \neq ermit$).

Today on a literature lesson Vera learned that quatrains can contain four different schemes of rhymes, namely the following ones (the same letters stand for rhyming lines):

- Clerihew ($aabb$);
- Alternating ($abab$);
- Enclosed ($abba$).

If all lines of a quatrain pairwise rhyme, then the quatrain can belong to any rhyme scheme (this situation is represented by $aaaa$).

If all quatrains of a poem belong to the same rhyme scheme, then we can assume that the whole poem belongs to this rhyme scheme. If in each quatrain all lines pairwise rhyme, then the rhyme scheme of the poem is $aaaa$. Let us note that it doesn't matter whether lines from different quatrains rhyme with each other or not. In other words, it is possible that different quatrains aren't connected by a rhyme.

Vera got a long poem as a home task. The girl has to analyse it and find the poem rhyme scheme. Help Vera cope with the task.

Input

The first line contains two integers n and k ($1 \le n \le 2500$, $1 \le k \le 5$) — the number of quatrains in the poem and the vowel's number, correspondingly. Next $4n$ lines contain the poem. Each line is not empty and only consists of small Latin letters. The total length of the lines does not exceed 10^4.

If we assume that the lines are numbered starting from 1, then the first quatrain contains lines number $1, 2, 3, 4$; the second one contains lines number $5, 6, 7, 8$; and so on.

Output

Print the rhyme scheme of the poem as "`aabb`", "`abab`", "`abba`", "`aaaa`"; or "`NO`" if the poem does not belong to any of the above mentioned schemes.

Sample test(s)

input

```
1 1
day
may
sun
fun
```

output

```
aabb
```

input

```
1 1
day
may
gray
way
```

output

```
aaaa
```

input

```
2 1
a
a
a
a
```

```
a
a
e
e
```

output

```
aabb
```

input

```
2 1
day
may
sun
fun
test
hill
fest
thrill
```

output

```
NO
```

Note

In the last sample both quatrains have rhymes but finding the common scheme is impossible, so the answer is "`NO`".

B. Digits Permutations

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

Andrey's favourite number is n. Andrey's friends gave him two identical numbers n as a New Year present. He hung them on a wall and watched them adoringly.

Then Andrey got bored from looking at the same number and he started to swap digits first in one, then in the other number, then again in the first number and so on (arbitrary number of changes could be made in each number). At some point it turned out that if we sum the resulting numbers, then the number of zeroes with which the sum will end would be maximum among the possible variants of digit permutations in those numbers.

Given number n, can you find the two digit permutations that have this property?

Input

The first line contains a positive integer n — the original number. The number of digits in this number does not exceed 10^5. The number is written without any leading zeroes.

Output

Print two permutations of digits of number n, such that the sum of these numbers ends with the maximum number of zeroes. The permutations can have leading zeroes (if they are present, they all should be printed). The permutations do not have to be different. If there are several answers, print any of them.

Sample test(s)

input

```
198
```

output

```
981
819
```

input

```
500
```

output

```
500
500
```

C. Mushroom Gnomes - 2

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

One day Natalia was walking in the woods when she met a little mushroom gnome. The gnome told her the following story:

Everybody knows that the mushroom gnomes' power lies in the magic mushrooms that grow in the native woods of the gnomes. There are n trees and m magic mushrooms in the woods: the i-th tree grows at a point on a straight line with coordinates a_i and has the height of h_i, the j-th mushroom grows at the point with coordinates b_j and has magical powers z_j.

But one day wild mushroommunchers, the sworn enemies of mushroom gnomes unleashed a terrible storm on their home forest. As a result, some of the trees began to fall and crush the magic mushrooms. The supreme oracle of mushroom gnomes calculated in advance the probability for each tree that it will fall to the left, to the right or will stand on. If the tree with the coordinate x and height h falls to the left, then all the mushrooms that belong to the right-open interval $[x - h, x)$, are destroyed. If a tree falls to the right, then the mushrooms that belong to the left-open interval $(x, x + h]$ are destroyed. Only those mushrooms that are not hit by a single tree survive.

Knowing that all the trees fall independently of each other (i.e., all the events are mutually independent, and besides, the trees do not interfere with other trees falling in an arbitrary direction), the supreme oracle was also able to quickly calculate what would be the expectation of the total power of the mushrooms which survived after the storm. His calculations ultimately saved the mushroom gnomes from imminent death.

Natalia, as a good Olympiad programmer, got interested in this story, and she decided to come up with a way to quickly calculate the expectation of the sum of the surviving mushrooms' power.

Input

The first line contains two integers n and m ($1 \le n \le 10^5$, $1 \le m \le 10^4$) — the number of trees and mushrooms, respectively.

Each of the next n lines contain four integers — a_i, h_i, l_i, r_i ($|a_i| \le 10^9$, $1 \le h_i \le 10^9$, $0 \le l_i, r_i, l_i + r_i \le 100$) which represent the coordinate of the i-th tree, its height, the percentage of the probabilities that the tree falls to the left and to the right, respectively (the remaining percentage is the probability that the tree will stand on).

Each of next m lines contain two integers b_j, z_j ($|b_j| \le 10^9$, $1 \le z_j \le 10^3$) which represent the coordinate and the magical power of the j-th mushroom, respectively.

An arbitrary number of trees and mushrooms can grow in one point.

Output

Print a real number — the expectation of the total magical power of the surviving mushrooms. The result is accepted with relative or absolute accuracy 10^{-4}.

Sample test(s)

input

```
1 1
2 2 50 50
1 1
```

output

```
0.5000000000
```

input

```
2 1
2 2 50 50
4 2 50 50
3 1
```

output

```
0.2500000000
```

Note

It is believed that the mushroom with the coordinate x belongs to the right-open interval $[l, r)$ if and only if $l \le x < r$. Similarly, the mushroom with the coordinate x belongs to the left-open interval $(l, r]$ if and only if $l < x \le r$.

In the first test the mushroom survives with the probability of 50%, depending on where the single tree falls.

In the second test the mushroom survives only if neither of the two trees falls on it. It occurs with the probability of 50% × 50% = 25%.

Pretest №12 is the large test with 10^5 trees and one mushroom.

D. World of Darkraft

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

Recently Roma has become the happy owner of a new game World of Darkraft. This game combines elements of virtually all known genres, and on one of the later stages of the game Roma faced difficulties solving a puzzle.

In this part Roma fights with a cunning enemy magician. The battle takes place on a rectangular field plaid $n \times m$. Each cell contains one magical character: `L`, `R` or `X`. Initially all the squares of the field are "active".

The players, Roma and enemy magician, take turns. Roma makes the first move. During a move a player selects one of the active cells. Then depending on the image in the character in the cell one of the following actions takes place:

- `L` — magical waves radiate from the cell to the left downwards and to the right upwards along diagonal paths. All cells on the path of the waves (including the selected cell too) become inactive. The waves continue until the next inactive cell or to the edge of the field if there are no inactive cells on the way.
- `R` — the magical waves radiate to the left upwards and to the right downwards.
- `X` — the magical waves radiate in all four diagonal directions.

If the next player cannot make a move (i.e., all cells are inactive), he loses.

Roma has been trying to defeat the computer opponent for three days but he just keeps losing. He asks you to help him and determine whether it is guaranteed that he can beat the opponent, or he will have to hack the game.

Input

The first line contains two space-separated integers n and m $(1 \le n, m \le 20)$.

Next n lines contain m characters describing the playing field: the j-th character of the i-th line equals to the magical character of the corresponding field square.

Output

On the first line print "`WIN`" if Roma can win or "`LOSE`" if it is impossible to win considering that the opponent pays optimally.

Sample test(s)

input

```
2 2
RL
LR
```

output

```
LOSE
```

input

```
2 2
RR
RR
```

output

```
WIN
```

Note

In the first test each move makes one diagonal line of the square inactive, thus it is guaranteed that Roma loses after two moves.

There are three variants of making a move in the second test: to "finish off" the main diagonal line or any of the squares that are left. That means that after three moves the game stops and Roma wins.

E. Hellish Constraints

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Katya recently started to invent programming tasks and prepare her own contests. What she does not like is boring and simple constraints. Katya is fed up with all those "N does not exceed a thousand" and "the sum of a_i does not exceed a million" and she decided to come up with something a little more complicated.

The last problem written by Katya deals with strings. The input is a string of small Latin letters. To make the statement longer and strike terror into the people who will solve the contest, Katya came up with the following set of k restrictions of the same type (characters in restrictions can be repeated and some restrictions may contradict each other):

- The number of characters c_1 in a string is not less than l_1 and not more than r_1.
- ...
- The number of characters c_i in a string is not less than l_i and not more than r_i.
- ...
- The number of characters c_k in a string is not less than l_k and not more than r_k.

However, having decided that it is too simple and obvious, Katya added the following condition: a string meets no less than L and not more than R constraints from the above given list.

Katya does not like to compose difficult and mean tests, so she just took a big string s and wants to add to the tests all its substrings that meet the constraints. However, Katya got lost in her conditions and asked you to count the number of substrings of the string s that meet the conditions (each occurrence of the substring is counted separately).

Input

The first line contains a non-empty string s, consisting of small Latin letters. The length of the string s does not exceed 10^5.

The second line contains three space-separated integers k, L and R ($0 \le L \le R \le k \le 500$).

Next k lines contain Katya's constrictions in the following form "c_i l_i r_i". All letters c_i are small Latin letters, l_i and r_i are integers ($0 \le l_i \le r_i \le |s|$, where $|s|$ is the length of string s). Letters c_i are not necessarily different.

Output

Print a single number — the number of substrings that meet the constrictions.

Please do not use the `%lld` specificator to read or write 64-bit integers in C++. It is preferred to use the `cout` stream or the `%I64d` specificator.

Sample test(s)

input

```
codeforces
2 0 0
o 1 2
e 1 2
```

output

```
7
```

input

```
codeforces
2 1 1
o 1 2
o 1 2
```

output

```
0
```

Note

In the first test we should count the number of strings that do not contain characters "`e`" and "`o`". All such strings are as follows (in the order of occurrence in the initial string from the left to the right): "`c`", "`d`"', "`f`", "`r`", "`rc`", "`c`", "`s`".

In the second test we cannot achieve fulfilling exactly one of the two identical constrictions, so the answer is 0.