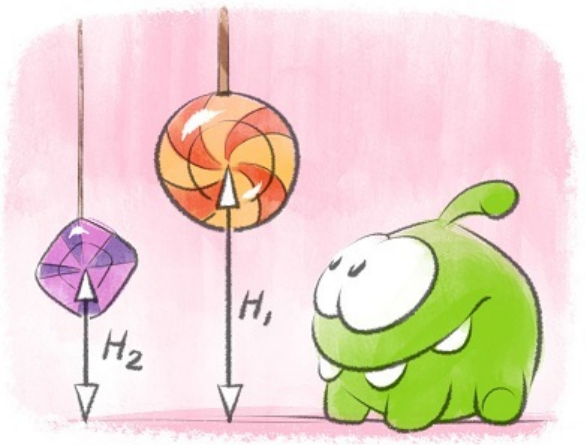
Zepto Code Rush 2014

A. Feed with Candy

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

The hero of the Cut the Rope game is a little monster named Om Nom. He loves candies. And what a coincidence! He also is the hero of today's problem.



One day, Om Nom visited his friend Evan. Evan has n candies of two types (fruit drops and caramel drops), the i-th candy hangs at the height of h_i centimeters above the floor of the house, its mass is m_i. Om Nom wants to eat as many candies as possible. At the beginning Om Nom can make at most x centimeter high jumps. When Om Nom eats a candy of mass y, he gets stronger and the height of his jump increases by y centimeters.

What maximum number of candies can Om Nom eat if he never eats two candies of the same type in a row (Om Nom finds it too boring)?

Input

The first line contains two integers, n and x $(1 \le n, x \le 2000)$ — the number of sweets Evan has and the initial height of Om Nom's jump.

Each of the following n lines contains three integers t_i, h_i, m_i $(0 \le t_i \le 1;\ 1 \le h_i, m_i \le 2000)$ — the type, height and the mass of the i-th candy. If number t_i equals 0, then the current candy is a caramel drop, otherwise it is a fruit drop.

Output

Print a single integer — the maximum number of candies Om Nom can eat.

Sample test(s)

input
5 3 0 2 4 1 3 1 0 8 3 0 20 10 1 5 5
output
4

Note

One of the possible ways to eat 4 candies is to eat them in the order: $1, 5, 3, 2$. Let's assume the following scenario:

1. Initially, the height of Om Nom's jump equals 3. He can reach candies 1 and 2. Let's assume that he eats candy 1. As the mass of this candy equals 4, the height of his jump will rise to $3 + 4 = 7$.
2. Now Om Nom can reach candies 2 and 5. Let's assume that he eats candy 5. Then the height of his jump will be $7 + 5 = 12$.
3. At this moment, Om Nom can reach two candies, 2 and 3. He won't eat candy 2 as its type matches the type of the previously eaten candy. Om Nom eats candy 3, the height of his jump is $12 + 3 = 15$.
4. Om Nom eats candy 2, the height of his jump is $15 + 1 = 16$. He cannot reach candy 4.

B. Om Nom and Spiders

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

Om Nom really likes candies and doesn't like spiders as they frequently steal candies. One day Om Nom fancied a walk in a park. Unfortunately, the park has some spiders and Om Nom doesn't want to see them at all.

The park can be represented as a rectangular $n \times m$ field. The park has k spiders, each spider at time 0 is at some cell of the field. The spiders move all the time, and each spider always moves in one of the four directions (left, right, down, up). In a unit of time, a spider crawls from his cell to the side-adjacent cell in the corresponding direction. If there is no cell in the given direction, then the spider leaves the park. The spiders do not interfere with each other as they move. Specifically, one cell can have multiple spiders at the same time.

Om Nom isn't yet sure where to start his walk from but he definitely wants:

- to start walking at time 0 at an upper row cell of the field (it is guaranteed that the cells in this row do not contain any spiders);
- to walk by moving down the field towards the lowest row (the walk ends when Om Nom leaves the boundaries of the park).

We know that Om Nom moves by jumping. One jump takes one time unit and transports the little monster from his cell to either a side-adjacent cell on the lower row or outside the park boundaries.

Each time Om Nom lands in a cell he sees all the spiders that have come to that cell at this moment of time. Om Nom wants to choose the optimal cell to start the walk from. That's why he wonders: for each possible starting cell, how many spiders will he see during the walk if he starts from this cell? Help him and calculate the required value for each possible starting cell.

Input

The first line contains three integers n, m, k $(2 \le n, m \le 2000; 0 \le k \le m(n - 1))$.

Each of the next n lines contains m characters — the description of the park. The characters in the i-th line describe the i-th row of the park field. If the character in the line equals "`.`", that means that the corresponding cell of the field is empty; otherwise, the character in the line will equal one of the four characters: "`L`" (meaning that this cell has a spider at time 0, moving left), "`R`" (a spider moving right), "`U`" (a spider moving up), "`D`" (a spider moving down).

It is guaranteed that the first row doesn't contain any spiders. It is guaranteed that the description of the field contains no extra characters. It is guaranteed that at time 0 the field contains exactly k spiders.

Output

Print m integers: the j-th integer must show the number of spiders Om Nom will see if he starts his walk from the j-th cell of the first row. The cells in any row of the field are numbered from left to right.

Sample test(s)

input
3 3 4 ... R.L R.U
output
0 2 2

input
2 2 2 .. RL
output
1 1

input

```
2 2 2
..
LR
```

output

```
0 0
```

input

```
3 4 8
....
RRLL
UUUU
```

output

```
1 3 3 1
```

input

```
2 2 2
..
UU
```

output

```
0 0
```

Note

Consider the first sample. The notes below show how the spider arrangement changes on the field over time:

```
...        ...        ..U        ...
R.L   ->   .*U   ->   L.R   ->   ...
R.U        .R.        ..R        ...
```

Character "*" represents a cell that contains two spiders at the same time.

- If Om Nom starts from the first cell of the first row, he won't see any spiders.
- If he starts from the second cell, he will see two spiders at time 1.
- If he starts from the third cell, he will see two spiders: one at time 1, the other one at time 2.

C. Dungeons and Candies

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

During the loading of the game "Dungeons and Candies" you are required to get descriptions of k levels from the server. Each description is a map of an $n \times m$ checkered rectangular field. Some cells of the field contain candies (each cell has at most one candy). An empty cell is denoted as "." on the map, but if a cell has a candy, it is denoted as a letter of the English alphabet. A level may contain identical candies, in this case the letters in the corresponding cells of the map will be the same.

When you transmit information via a network, you want to minimize traffic — the total size of the transferred data. The levels can be transmitted in any order. There are two ways to transmit the current level A:

1. You can transmit the whole level A. Then you need to transmit $n \cdot m$ bytes via the network.
2. You can transmit the difference between level A and some previously transmitted level B (if it exists); this operation requires to transmit $d_{A,B} \cdot w$ bytes, where $d_{A,B}$ is the number of cells of the field that are different for A and B, and w is a constant. Note, that you should compare only the corresponding cells of levels A and B to calculate $d_{A,B}$. You cannot transform the maps of levels, i.e. rotate or shift them relatively to each other.

Your task is to find a way to transfer all the k levels and minimize the traffic.

Input

The first line contains four integers n, m, k, w $(1 \le n, m \le 10;\ 1 \le k, w \le 1000)$. Then follows the description of k levels. Each level is described by n lines, each line contains m characters. Each character is either a letter of the English alphabet or a dot (".). Please note that the case of the letters matters.

Output

In the first line print the required minimum number of transferred bytes.

Then print k pairs of integers $x_1, y_1, x_2, y_2, \ldots, x_k, y_k$, describing the way to transfer levels. Pair x_i, y_i means that level x_i needs to be transferred by way y_i. If y_i equals 0, that means that the level must be transferred using the first way, otherwise y_i must be equal to the number of a previously transferred level. It means that you will transfer the difference between levels y_i and x_i to transfer level x_i. Print the pairs in the order of transferring levels. The levels are numbered 1 through k in the order they follow in the input.

If there are multiple optimal solutions, you can print any of them.

Sample test(s)

input

```
2 3 3 2
A.A
...
A.a
..C
X.Y
...
```

output

```
14
1 0
2 1
3 1
```

input

```
1 1 4 1
A
.
B
.
```

output

```
3
1 0
2 0
4 2
3 0
```

input

```
1 3 5 2
ABA
BBB
BBA
BAB
ABB
```

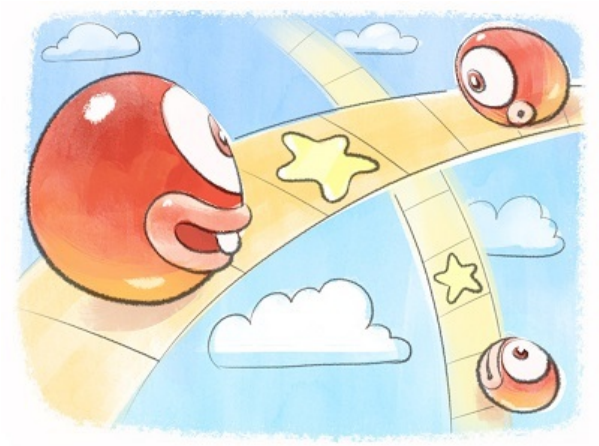
output

```
11
1 0
3 1
2 3
4 2
5 1
```

D. Pudding Monsters

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Have you ever played Pudding Monsters? In this task, a simplified one-dimensional model of this game is used.



Imagine an infinite checkered stripe, the cells of which are numbered sequentially with integers. Some cells of the strip have monsters, other cells of the strip are empty. All monsters are made of pudding, so if there are two monsters in the neighboring cells, they stick to each other (literally). Similarly, if several monsters are on consecutive cells, they all stick together in one block of monsters. We will call the stuck together monsters a block of monsters. A detached monster, not stuck to anyone else, is also considered a block.

In one move, the player can take any block of monsters and with a movement of his hand throw it to the left or to the right. The selected monsters will slide on until they hit some other monster (or a block of monsters).

For example, if a strip has three monsters in cells 1, 4 and 5, then there are only four possible moves: to send a monster in cell 1 to minus infinity, send the block of monsters in cells 4 and 5 to plus infinity, throw monster 1 to the right (it will stop in cell 3), throw a block of monsters in cells 4 and 5 to the left (they will stop in cells 2 and 3).

Some cells on the strip are marked with stars. These are the special cells. The goal of the game is to make the largest possible number of special cells have monsters on them.

You are given the numbers of the special cells on a strip as well as the initial position of all monsters. What is the maximum number of special cells that will contain monsters in the optimal game?

Input

The first line contains two integers n and m $(1 \le n \le 10^5; 1 \le m \le 2000)$ — the number of monsters on the strip and the number of special cells.

The second line contains n distinct integers — the numbers of the cells with monsters, then the third line contains m distinct integers — the numbers of the special cells. It is guaranteed that all the numbers of the cells are positive integers not exceeding $2 \cdot 10^5$.

Output

Print a single integer — the maximum number of special cells that will contain monsters in the optimal game.

Sample test(s)

input
3 2 1 3 5 2 4
output
2

input
4 2 1 3 4 6 2 5
output
2

input
4 2 1 8 4 5 7 2
output

```
1
```

E. Cardboard Box

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Everyone who has played Cut the Rope knows full well how the gameplay is organized. All levels in the game are divided into boxes. Initially only one box with some levels is available. Player should complete levels to earn stars, collecting stars opens new box with levels.

Imagine that you are playing Cut the Rope for the first time. Currently you have only the levels of the first box (by the way, it is called "Cardboard Box"). Each level is characterized by two integers: a_i — how long it takes to complete the level for one star, b_i — how long it takes to complete the level for two stars $(a_i < b_i)$.

You want to open the next box as quickly as possible. So, you need to earn at least w stars. How do make it happen? Note that the level can be passed only once: either for one star or for two. You do not necessarily need to pass all the levels.

Input

The first line contains two integers n and w $(1 \le n \le 3 \cdot 10^5;\ 1 \le w \le 2n)$ — the number of levels in the first box and the number of stars you need to open another box. Each of the following n lines contains two integers a_i and b_i $(1 \le a_i < b_i \le 10^9)$ — the attributes of the i-th level.

Output

In the first line print integer t — the minimum time you need to open the next box.

In the next line, print n digits without spaces — the description of the optimal scenario:

- if you need to pass the i-th level for one star, the i-th digit should equal 1;
- if you need to pass the i-th level for two stars, the i-th digit should equal 2;
- if you do not need to pass the i-th level at all, the i-th digit should equal 0.

Sample test(s)

input
2 3 1 2 1 2
output
3 12

input
5 3 10 20 5 10 10 20 6 9 25 30
output
14 01020

Note

In the first test sample, answer `21` is also assumed correct.

F. Banners

time limit per test: 5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

All modern mobile applications are divided into free and paid. Even a single application developers often release two versions: a paid version without ads and a free version with ads.

Suppose that a paid version of the app costs p (p is an integer) rubles, and the free version of the application contains c ad banners. Each user can be described by two integers: a_i — the number of rubles this user is willing to pay for the paid version of the application, and b_i — the number of banners he is willing to tolerate in the free version.

The behavior of each member shall be considered strictly deterministic:

- if for user i, value b_i is at least c, then he uses the free version,
- otherwise, if value a_i is at least p, then he buys the paid version without advertising,
- otherwise the user simply does not use the application.

Each user of the free version brings the profit of $c \times w$ rubles. Each user of the paid version brings the profit of p rubles.

Your task is to help the application developers to select the optimal parameters p and c. Namely, knowing all the characteristics of users, for each value of c from 0 to $(max\ b_i) + 1$ you need to determine the maximum profit from the application and the corresponding parameter p.

Input

The first line contains two integers n and w ($1 \le n \le 10^5$; $1 \le w \le 10^5$) — the number of users and the profit from a single banner. Each of the next n lines contains two integers a_i and b_i ($0 \le a_i, b_i \le 10^5$) — the characteristics of the i-th user.

Output

Print $(max\ b_i) + 2$ lines, in the i-th line print two integers: pay — the maximum gained profit at $c = i$ - 1, p ($0 \le p \le 10^9$) — the corresponding optimal app cost. If there are multiple optimal solutions, print any of them.

Sample test(s)

input

```
2 1
2 0
0 2
```

output

```
0 3
3 2
4 2
2 2
```

input

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

output

```
0 4
3 4
7 3
7 2
4 2
```