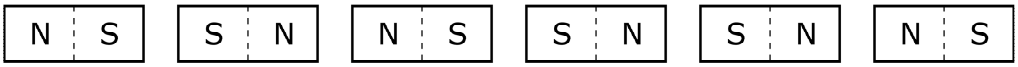
TASK	MAGNETI	KARTA	SLIKA
input	standard input		
output	standard output		
time limit	1 second		
memory limit	64 MB		
points	**60**	**60**	**80**
	200		

The crazy scientist Matija is performing crazy experiments with even crazier magnets. Each magnet looks like a small stick. It is 1 unit long and has two magnetic poles, South (S) and North (N). Poles of two magnets with the same orientation cannot be joined, but **opposite poles attract** each other and make the magnets join. The magnets are so strong that even Chuck Norris cannot separate them once they have joined.

Before the experiment starts, Matija puts N magnets on the floor of his laboratory, like this:



Example arrangement of the magnets on the floor. First sample test case on the next page.

Before Matija can proceed, opposite poles of neighbouring magnets attrat each other and those magnets join, forming a single inseparable magnet.

The magnets after the initial joining.

He noticed that by flipping a magnet in the new sequence, he can cause more magnets to join. He decided to create a magnet **exactly L units long**. Write a program that determines the **smallest number of flip operations** needed to get a magnet of length L.

INPUT

The first line of input contains two integers N and L ($1 \leq N \leq 500\,000$, $1 \leq L \leq N$), the number of starting magnets and the required length. The starting magnets are 1 unit long.

The next line contains N strings "NS" or "SN", separated by single spaces. These are the orientations of the magnets after Matija puts them on the floor, but before the initial joining.

The input data will be such that a solution will always exist.

Note: in 70% of the official test data, N will be less than 1000.

OUTPUT

Output the smallest number of flips needed.

EXAMPLE TEST CASES

input	**input**	**input**
6 4	4 4	15 13
NS SN NS SN SN NS	NS SN NS SN	SN NS NS SN NS SN SN NS NS SN SN NS NS NS SN
output	**output**	**output**
1	2	3

Clarification for first sample: the fourth and fifth magnets immediately join. If Matija flips that 2 unit long magnet, it will connect to both neighbouring magnets, forming a magnet of length 4.

A geographical map is represented with R×C characters.

There are towns on the map. The **position** of a town is marked with the lowercase letter 'x', and the **name** of the town is a sequence of consecutive uppercase letters read from left to right. Each town's name is next to its position. More precisely, at least one letter in a town's name is in one of the 8 squares around the town's position.

There may be multiple names surrounding a town's position and, similarly, there may be multiple town positions adjacent to a name. However, it is guaranteed that it is possible to **uniquely determine the positions and names** for all towns.

Write a program which **determines the positions and names** of all towns.

INPUT

The first line of input consists of two integers, R and C ($1 \leq R \leq 50$, $1 \leq S \leq 50$), the numbers of rows and columns.

Each of the next R lines contains C characters, the map. Each character is either '.', 'x', or one of the uppercase letters of the English alphabet.

The town names will be unique. Town names in the same row will be separated by at least one '.' or 'x' character.

OUTPUT

For each town, output a line containing its position on the map and its name. The rows are numbered 1 to R (top to bottom) and the columns are numbered 1 to C (left to right). The towns can be printed in any order.

EXAMPLE TEST CASES

input

```
12 31
...............................
.............ZAGREB............
.............x.................
.......................OSIJEKx.
.....x.........................
......RIJEKA...................
...............................
...............................
...............................
.............SPLIT.............
.............x.................
...............................
```

output

```
3 15 ZAGREB
4 30 OSIJEK
5 6 RIJEKA
11 16 SPLIT
```

input

```
3 8
.VELIKI.
x.....x.
..MALI..
```

output

```
2 1 VELIKI
2 7 MALI
```

Mirko was bored during computer class, so he decided to open "Paint" and have some fun. He created a new image N×N pixels in size.

Because he wasn't paying attention when the teacher was explaining how to use "Paint", Mirko only knows how to draw squares, of various colors. First he drew a square of color 1, then one of color 2 (which may have covered part of the first square) and so forth until square K.

When the teacher saw Mirko wasting his time, she immediately sent him to the principal. However, when she looked carefully at the image, she was intrigued by the unusual scrawl. She figured out the way the image was drawn. However, she couldn't figure out the initial lengths of the squares' sizes.

She deduced that, **for each color A**, it is possible to find an interval [a, b] such that for each x in that interval, it is possible that **the length of the sides of square A could be x**.

Write a program which helps Mirko's teacher find **the largest associated interval for each color**.

Mirku je dosadno na satu informatike pa se odlučio malo zabaviti crtanjem. Otvorio je "Paint" i u njemu novu sliku dimenzije N×N piksela.

INPUT

The first line of input contains two integers N and K ($1 \leq N \leq 1\,000$, $1 \leq K \leq 9$), the dimensions of the image and the number of squares Mirko drew.

Each of following N lines contains N characters. Those characters are either '.' or one of the digits 1 to K. The '.' character means that no square ever covered the pixel. The digit x means that the last square to cover that pixel was one with the color x.

OUTPUT

The output should contain K lines. In each of the lines you should output two integers. The two integers on line i represent the smallest and largest possible lengths of the side of square i.

EXAMPLE TEST CASES

input

```
10 4
..........
..........
.....33...
...1133...
...12222..
..444422..
..444422..
..444422..
..4444....
..........
```

output

```
2 4
4 4
2 2
4 4
```

input

```
5 3
..333
.1333
.1333
.1111
.1111
```

output

```
4 4
1 3
3 3
```

input

```
7 5
...555.
...555.
..2555.
.44443.
.4444..
.4444..
.4444..
```

output

```
1 4
1 3
1 3
4 4
3 3
```