Task	PRASE	MAGIJA	MARATON	KAMEN	V	PROSTOR
Input	standard input (keyboard)					
Output	standard output (screen)					
Memory limit (heap)	32 MB	32 MB	32 MB	32 MB	32 MB	32 MB
Memory limit (stack)	8 MB	8 MB	8 MB	8 MB	8 MB	8 MB
Time limit (per test)	1 second	1 second	1 second	1 second	1 second	2 seconds
Number of tests	5	10	10	10	14	10
Points per test	4	3	3	5	5	10
Total points	**20**	**30**	**30**	**50**	**70**	**100**
	300					

Note: The evaluation system has two Intel Pentium 4 3.0 GHz processors and is running the Linux operating system. The following compile options are used for different languages:

- C: –O2 –s –static –std=c99 –lm
- C++: –O2 –s –static –lm
- Pascal: –O1 –XS

N children are eating lunch at the table. Children take turns in taking food from the table.

Some of the children haven't yet been taught proper manners so they jump at the food without giving the others a chance. If at any point a child takes a piece of food, and that child **had already taken more food than the other children all together** (not including the new piece of food), then the mother will warn that child to behave.

You will be given the order in which the children take food. Write a program that calculates how many times the mother has to warn the children.

Input

The first line of input contains an integer N (1 ≤ N ≤ 100), how many pieces of food the children take.

Each of the following N lines contains the name of a child that took one piece of food. The names will be strings of at most 20 lowercase letters of the English alphabet.

Output

Output the number of warnings on a single line.

Sample test data

input

```
4
mirko
stanko
stanko
stanko
```

output

```
1
```

input

```
17
a
b
b
a
a
a
c
a
b
b
c
b
b
b
b
b
b
```

output

```
4
```

2. MAGIJA

The well-known magician Al'Dimi Kartimi needs a program to help him design the back side of his cards.

Al'Dimi first draws the upper left quarter of the card, mirrors it horizontally onto the upper right quarter and then vertically mirrors the entire upper half onto the lower half.

After the mirroring, Al'Dimi also adds a small error (changes the appearance of one square) to help him determine which card it is (to cheat, if you will).

Help Al'Dimi by writing a program that, given the design of the upper left quarter and the location of the error, draws the entire back side.

Here are three examples of Al'Dimi's cards (the error is shaded gray):

```
###.##.###
##########
.########.
..######..
####.#####
##########
..######..
.########.
##########
###.##.###
```

```
#.#..#.#
#.####.#
#.####.#
........
.#.##.#.
.#.##.#.
........
#.####.#
#.####.#
#.#.##.#
```

```
.#.##.##
#.#..#.#
........
..#..#..
..#..#..
........
#.#..#.#
##.##.##
```

Input

The first line of input contains two integers, R and C (1 ≤ R, C ≤ 50), the number of rows and columns in the upper left quarter of the card.

Each of the R following lines contains C characters '.' or '#', the design of the upper left quarter.

The next line contains two integers, A and B (1 ≤ A ≤ 2R, 1 ≤ B ≤ 2C), the row and column of the error.

Output

Output 2R rows, each containing 2C characters, the design of the back side.

Sample test data

input

```
2 2
#.
.#
3 3
```

output

```
#..#
.##.
.#..
#..#
```

input

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

output

```
###.##
######
######
######
######
######
```

input

```
5 4
#.#.
#.##
#.##
....
.#.#
10 5
```

output

```
#.#..#.#
#.####.#
#.####.#
........
.#.##.#.
.#.##.#.
........
#.####.#
#.####.#
#.#.##.#
```

3. MARATON

Albert, Barbara, Casper, Dinko, Eustahije are starting a marathon game of tic-tac-toe, played on an N×N board.

Initially, all squares on the board are empty and players take turns writing the first letter of their name into any of the empty squares (because the players are elite, no two players have the same first letter).

The game ends when some player **places 3 of his or her letters consecutively** in a row, column or diagonally. That player is declared the winner.

Write a program that, given the state of the board, determines if the game is over and who won if it is.

Input

The first line of input contains the integer N (1 ≤ N ≤ 30), the size of the board.

The following N lines contain N characters each. The characters will be uppercase letters of the English alphabet or '.' (if the square is empty).

The input data will be such that there is at most one winner.

Output

If the game is over, output the first letter of the winner's name. If not, output "ongoing" (even if the board is full).

Sample test data

input

```
3
XOC
XOC
X..
```

output

```
X
```

input

```
4
....
..A.
AAB.
.B.B
```

output

```
ongoing
```

input

```
3
ABB
AAA
BBA
```

output

```
A
```

4. KAMEN

For nearly two weeks now, Domeniko has been lying in his bed because his friend Nedjeljko accidently threw a large rock on his left foot. As Domeniko has already solved the tasks from all Croatian national competitions since 1998, he has to find a new way to kill time.

Domeniko's new game is played on an R×C board. Initially every square is either empty or blocked by a wall. Domeniko throws a rock into the board by putting it in the topmost row of a column and then letting gravity do the rest.

Gravity works as follows:

- If the square under the rock is a wall or if the rock is in the bottom row of a column, then the rock remains there.
- If the square under the rock is empty, then the rock moves to that square.
- If the square under the rock contains another rock, then the falling rock may slide sideways:
    - If the squares left and left-down of the rock are empty, then the rock slides one square left.
    - If the rock doesn't slide left and the squares to the right and right-down are empty, then the rock slides one square right.
    - Otherwise, the rock remains there and never moves again.

Domeniko will never throw another rock while the previous rock hasn't settled down.

Write a program that draws the board after Domeniko throws all his rocks into the board, if we know the columns that Domeniko threw his rocks into, in order.

Note: Domeniko will never throw a rock into column in which the top row isn't empty.

Input

The first line contains integers R and C ($1 \le R \le 30000$, $1 \le C \le 30$), the size of the board.

Each of the following R lines contains C characters, the initial layout of the board. A '.' represents an empty field, while the uppercase letter 'X' is a square blocked by a wall.

The next line contains an integer N ($1 \le N \le 100000$), the number of rocks Domeniko throws.

Each of the following N lines contains an integer between 1 and C, the column into which Domeniko throws a rock (the leftmost column is column 1).

Note: In 60% of the test cases, R will be no more than 30.

Output

Output R lines, each containing C characters, the final layout of the board. Rocks should be presented with an uppercase letter 'O'.

Sample test data

input

```
5 4
....
....
X...
....
....
4
1
1
1
1
```

output

```
....
O...
X...
....
OOO.
```

input

```
7 6
......
......
...XX.
......
......
.XX...
......
6
1
4
4
6
4
4
```

output

```
......
...O..
...XX.
......
.OO...
.XX...
O..O.O
```

In the first example, all rocks are thrown in the first column. The first rock stops on the wall. The second rock falls on the first, slides right and stops at the bottom of the second column. The third rock falls on the first then on the second rock, slides left and rests at the bottom of the first column. The fourth rock falls on the first then on the second, then slides right.

5. V

Zvonko is playing with digits again, even though his mother has warned him that he is doing too much math and should go outside to play with his friends.

In his latest game, Zvonko looks for **multiples** of an integer X, **composed only of certain digits**. A multiple of X is any number divisible by X.

In order to ruin Zvonko's fun, his mother decided to get a program that solves the problem. Write a program that calculates how many multiples of X are between A and B (inclusive), such that, when written in decimal, they contain only certain allowed digits.

Input

The first line of input contains three integers X, A and B ($1 \le X < 10^{11}$, $1 \le A \le B < 10^{11}$).

The second line contains the allowed digits. The digits will be given with no spaces, sorted in increasing order and without duplicates.

Output

Output the number of multiples Zvonko can make on a single line.

Sample test data

input

```
2 1 20
0123456789
```

output

```
10
```

input

```
6 100 9294
23689
```

output

```
111
```

input

```
5 4395 9999999999
12346789
```

output

```
0
```

6. PROSTOR

A long time ago in a three-dimensional space far away, a tribe of rectangles lived happily. The rectangles lived a spiritual life, parallel with one of the coordinate planes.

One day, a cuboid walked into their small world, riding steadily on an icosahedron, showing off its sharp corners and positive volume. The rectangles watched in awe and dreamed of being cuboids. Nothing would ever be the same from that day on. The rectangles started comparing each other by area, perimeter and even by the ratio of the lengths of their sides.

Soon the first conflict ensued over the ownership of shared points. In time, **each pair of rectangles sharing at least one point** (including those merely touching each other) got into a conflict and became enemies.

It is up to you to restore peace in the community, by meeting with every pair of rectangles in conflict. Write a program that finds **how many such pairs** there are.

Input

The first line of input contains the integer N (1 ≤ N ≤ 100 000), the number of rectangles.

Each of the following N lines contains 6 integers separated by single spaces. The first three numbers represent the coordinates of one corner of the rectangle, the other three are the coordinates of the opposite corner.

The coordinates are integers between 1 and 999 (inclusive).

Each rectangle is parallel to one of the coordinate planes, meaning that in exactly one of the three dimensions, the two corresponding coordinates will be equal.

Output

Output the total number of rectangles in conflict on a single line.

Sample test data

input

```
3
1 1 1 1 3 3
1 3 3 1 6 6
1 4 4 1 5 5
```

output

```
2
```

input

```
3
15 10 10 15 20 20
10 15 10 20 15 20
10 10 15 20 20 15
```

output

```
3
```

input

```
5
4 4 5 4 3 2
5 3 2 4 3 1
5 4 3 1 1 3
1 4 3 1 5 4
5 5 4 5 4 2
```

output

```
4
```