# A. Series of Crimes

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

The Berland capital is shaken with three bold crimes committed by the Pihsters, a notorious criminal gang.

The Berland capital's map is represented by an $n \times m$ rectangular table. Each cell of the table on the map represents some districts of the capital.

The capital's main detective Polycarpus took a map and marked there the districts where the first three robberies had been committed as asterisks. Deduction tells Polycarpus that the fourth robbery will be committed in such district, that all four robbed districts will form the vertices of some rectangle, parallel to the sides of the map.

Polycarpus is good at deduction but he's hopeless at math. So he asked you to find the district where the fourth robbery will be committed.

### Input

The first line contains two space-separated integers $n$ and $m$ $(2 \le n, m \le 100)$ — the number of rows and columns in the table, correspondingly.

Each of the next $n$ lines contains $m$ characters — the description of the capital's map. Each character can either be a "`.`" (dot), or an "`*`" (asterisk). A character equals "`*`" if the corresponding district has been robbed. Otherwise, it equals "`.`".

It is guaranteed that the map has exactly three characters "`*`" and we can always find the fourth district that meets the problem requirements.

### Output

Print two integers — the number of the row and the number of the column of the city district that is the fourth one to be robbed. The rows are numbered starting from one from top to bottom and the columns are numbered starting from one from left to right.

**Sample test(s)**

input

```
3 2
.*
..
**
```

output

```
1 1
```

input

```
3 3
*.*
*..
...
```

output

```
2 3
```

# B. Number of Triplets

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

You are given $n$ points on a plane. All points are different.

Find the number of different groups of three points $(A, B, C)$ such that point $B$ is the middle of segment $AC$.

The groups of three points are considered unordered, that is, if point $B$ is the middle of segment $AC$, then groups $(A, B, C)$ and $(C, B, A)$ are considered the same.

### Input

The first line contains a single integer $n$ $(3 \le n \le 3000)$ — the number of points.

Next $n$ lines contain the points. The $i$-th line contains coordinates of the $i$-th point: two space-separated integers $x_i, y_i$ $( - 1000 \le x_i, y_i \le 1000)$.

It is guaranteed that all given points are different.

### Output

Print the single number — the answer to the problem.

**Sample test(s)**

input

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

output

```
1
```

input

```
3
0 0
-1 0
0 1
```

output

```
0
```

# C. Trading Business

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

To get money for a new aeonic blaster, ranger Qwerty decided to engage in trade for a while. He wants to buy some number of items (or probably not to buy anything at all) on one of the planets, and then sell the bought items on another planet. Note that this operation is not repeated, that is, the buying and the selling are made only once. To carry out his plan, Qwerty is going to take a bank loan that covers all expenses and to return the loaned money at the end of the operation (the money is returned without the interest). At the same time, Querty wants to get as much profit as possible.

The system has $n$ planets in total. On each of them Qwerty can buy or sell items of $m$ types (such as food, medicine, weapons, alcohol, and so on). For each planet $i$ and each type of items $j$ Qwerty knows the following:

- $a_{ij}$ — the cost of buying an item;
- $b_{ij}$ — the cost of selling an item;
- $c_{ij}$ — the number of remaining items.

It is not allowed to buy more than $c_{ij}$ items of type $j$ on planet $i$, but it is allowed to sell any number of items of any kind.

Knowing that the hold of Qwerty's ship has room for no more than $k$ items, determine the maximum profit which Qwerty can get.

### Input

The first line contains three space-separated integers $n$, $m$ and $k$ ($2 \le n \le 10$, $1 \le m, k \le 100$) — the number of planets, the number of question types and the capacity of Qwerty's ship hold, correspondingly.

Then follow $n$ blocks describing each planet.

The first line of the $i$-th block has the planet's name as a string with length from $1$ to $10$ Latin letters. The first letter of the name is uppercase, the rest are lowercase. Then in the $i$-th block follow $m$ lines, the $j$-th of them contains three integers $a_{ij}$, $b_{ij}$ and $c_{ij}$ ($1 \le b_{ij} < a_{ij} \le 1000$, $0 \le c_{ij} \le 100$) — the numbers that describe money operations with the $j$-th item on the $i$-th planet. The numbers in the lines are separated by spaces.

It is guaranteed that the names of all planets are different.

### Output

Print a single number — the maximum profit Qwerty can get.

**Sample test(s)**

input

```
3 3 10
Venus
6 5 3
7 6 5
8 6 10
Earth
10 9 0
8 6 4
10 9 3
Mars
4 3 0
8 4 12
7 2 5
```

output

```
16
```

### Note

In the first test case you should fly to planet `Venus`, take a loan on 74 units of money and buy three items of the first type and 7 items of the third type ($3 \cdot 6 + 7 \cdot 8 = 74$). Then the ranger should fly to planet `Earth` and sell there all the items he has bought. He gets $3 \cdot 9 + 7 \cdot 9 = 90$ units of money for the items, he should give 74 of them for the loan. The resulting profit equals 16 units of money. We cannot get more profit in this case.

# D. Word Cut

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

Let's consider one interesting word game. In this game you should transform one word into another through special operations.

Let's say we have word $w$, let's split this word into two non-empty parts $x$ and $y$ so, that $w = xy$. A *split* operation is transforming word $w = xy$ into word $u = yx$. For example, a *split* operation can transform word "`wordcut`" into word "`cutword`".

You are given two words $start$ and $end$. Count in how many ways we can transform word $start$ into word $end$, if we apply exactly $k$ *split* operations consecutively to word $start$.

Two ways are considered different if the sequences of applied operations differ. Two operation sequences are different if exists such number $i$ ($1 \le i \le k$), that in the $i$-th operation of the first sequence the word splits into parts $x$ and $y$, in the $i$-th operation of the second sequence the word splits into parts $a$ and $b$, and additionally $x \ne a$ holds.

### Input

The first line contains a non-empty word $start$, the second line contains a non-empty word $end$. The words consist of lowercase Latin letters. The number of letters in word $start$ equals the number of letters in word $end$ and is at least $2$ and doesn't exceed $1000$ letters.

The third line contains integer $k$ ($0 \le k \le 10^5$) — the required number of operations.

### Output

Print a single number — the answer to the problem. As this number can be rather large, print it modulo $1000000007$ ($10^9 + 7$).

**Sample test(s)**

input

```
ab
ab
2
```

output

```
1
```

input

```
ababab
ababab
1
```

output

```
2
```

input

```
ab
ba
2
```

output

```
0
```

### Note

The sought way in the first sample is:

`ab` → `a|b` → `ba` → `b|a` → `ab`

In the second sample the two sought ways are:

- `ababab` → `abab|ab` → `ababab`
- `ababab` → `ab|abab` → `ababab`

# E. Playing with Superglue

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

Two players play a game. The game is played on a rectangular board with $n \times m$ squares. At the beginning of the game two different squares of the board have two chips. The first player's goal is to shift the chips to the same square. The second player aims to stop the first one with a *tube of superglue*.

We'll describe the rules of the game in more detail.

The players move in turns. The first player begins.

With every move the first player chooses one of his unglued chips, and shifts it one square to the left, to the right, up or down. It is not allowed to move a chip beyond the board edge. At the beginning of a turn some squares of the board may be covered with a glue. The first player can move the chip to such square, in this case the chip gets tightly glued and cannot move any longer.

At each move the second player selects one of the free squares (which do not contain a chip or a glue) and covers it with superglue. The glue dries long and squares covered with it remain sticky up to the end of the game.

If, after some move of the first player both chips are in the same square, then the first player wins. If the first player cannot make a move (both of his chips are glued), then the second player wins. Note that the situation where the second player cannot make a move is impossible — he can always spread the glue on the square from which the first player has just moved the chip.

We will further clarify the case where both chips are glued and are in the same square. In this case the first player wins as the game ends as soon as both chips are in the same square, and the condition of the loss (the inability to move) does not arise.

You know the board sizes and the positions of the two chips on it. At the beginning of the game all board squares are glue-free. Find out who wins if the players play optimally.

### Input

The first line contains six integers $n, m, x_1, y_1, x_2, y_2$ — the board sizes and the coordinates of the first and second chips, correspondingly ($1 \le n, m \le 100$; $2 \le n \times m$; $1 \le x_1, x_2 \le n$; $1 \le y_1, y_2 \le m$). The numbers in the line are separated by single spaces.

It is guaranteed that the chips are located in different squares.

### Output

If the first player wins, print "`First`" without the quotes. Otherwise, print "`Second`" without the quotes.

**Sample test(s)**

input

```
1 6 1 2 1 6
```

output

```
First
```

input

```
6 5 4 3 2 1
```

output

```
First
```

input

```
10 10 1 1 10 10
```

output

```
Second
```