1. PRASE

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 ($1 \leq N \leq 100$), 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

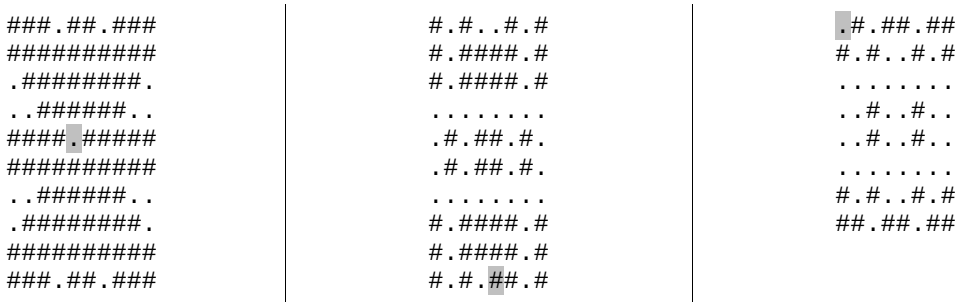
The well-known magician Al'Dimi Kartimi needs a program to help him design the back side of his cards.

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 ($1 \leq R, C \leq 50$), 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 ($1 \leq A \leq 2R$, $1 \leq B \leq 2C$), 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 ($1 \leq N \leq 30$), 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 ($1 \leq R \leq 30000$, $1 \leq C \leq 30$), 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 ($1 \leq N \leq 100000$), 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.

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 ($1 \leq X < 10^{11}$, $1 \leq A \leq B < 10^{11}$).

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	**input**	**input**
2 1 20 0123456789	6 100 9294 23689	5 4395 9999999999 12346789
output	**output**	**output**
10	111	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 ($1 \leq N \leq 100\,000$), 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
```

1. PARKING

Having dropped out of school because of chemistry, Luka got a job driving trucks. One evening he parked his three trucks in a rest area which charges for parking in an unusual way – they give a discount on quantity.

When only one truck is parked, the driver pays A kuna per minute. When two trucks are parked, the drivers each pay B kuna per minute. When three trucks are parked, the drivers each pay C kuna per minute.

Given the numbers A, B and C, as well as the intervals in which Luka's three trucks are parked, determines how much Luka needs to pay the owner of the rest area.

Input

The first line contains three integers A, B and C ($1 \leq C \leq B \leq A \leq 100$), the prices of parking as defined above.

Each of the following three lines contains two integers each. These are the arrival and departure times (in minutes) of one of Luka's trucks. The arrival time will always be earlier than the departure time. All time indexes will be between 1 and 100.

Output

Output the overall cost of Luka's parking his three trucks.

Sample test data

input

```
5 3 1
1 6
3 5
2 8
```

output

```
33
```

input

```
10 8 6
15 30
25 50
70 80
```

output

```
480
```

2. SEMAFORI

Luka is driving his truck along a long straight road with many traffic lights. For each traffic light he knows how long the red and green lights will be on (the cycle repeating endlessly).

When Luka starts his journey, all traffic lights are red and just started their cycle. Luka moves one distance unit per second. When a traffic light is red, he stops and waits until it turns green.

Write a program that determines how much time Luka needs to reach the end of the road. The start of the road is at distance zero, the end at distance L.

Input

The first line contains two integers N and L ($1 \leq N \leq 100$, $1 \leq L \leq 1000$), the number of traffic lights on the road and the length of the road.

Each of the next N lines contains three integers D, R and G, describing one traffic light ($1 \leq D < L$, $1 \leq R \leq 100$, $1 \leq G \leq 100$). D is the distance of the traffic light from the start of the road. R and G denote how long the red and green lights are on, respectively.

The traffic lights will be ordered in increasing order of D. No two traffic lights will share the same position.

Output

Output the time (in seconds) Luka needs to reach the end of the road.

Sample test data

input

```
2 10
3 5 5
5 2 2
```

output

```
12
```

input

```
4 30
7 13 5
14 4 4
15 3 10
25 1 1
```

output

```
36
```

In the first example, Luka will wait 2 seconds at the first traffic light. After that he will reach the second traffic light while it is green and be able to pass through immediately.

3. GRANICA

Luka started driving international routes with his truck. His biggest problem is the border with Slovenia. The border is a point of entrance into the European Union, so every truck is thoroughly examined. Because of this, Luka always has to wait several hours there. To kill the time, he comes up with various logic and math games.

In one of them, Luka first reads the numbers off of N license plates and writes them down on a piece of paper. Then he tries to find an integer M greater than 1 such that all integers on the paper give the same remainder when divided by M. Luka tries to find as many such integers M as possible.

Write a program that, given Luka's N integers, determines all such integers M.

Input

The first line contains the integer N ($2 \leq N \leq 100$), the number of integers on paper.

Each of the following N lines contains one integer between 1 and 1 000 000 000 (one billion). All these integers will be distinct.

The input data will guarantee that at least one integer M will always exist.

Output

Output all integers M separated by spaces, in any order.

Scoring

In test cases worth 60% points, each of the N numbers will be at most 10000.

Sample test data

input

```
3
6
34
38
```

output

```
2 4
```

input

```
5
5
17
23
14
83
```

output

```
3
```

In the first example, all integers give a remainder of 0 when divided by 2 and the remainder 2 when divided by 4.

4. GEORGE

Last week Mister George visited Croatia. Since Mister George is a very important person, while he was in a street, the police **disallowed entry** to that street, but vehicles that entered the street before Mister George could continue driving.

While Mister George was visiting, Luka drove his truck around town. But because of some of the streets being closed off, he couldn't make his delivery in time and almost lost his job. Although it is late now, he is wondering how he could have planned his delivery better i.e. what would have been the least time needed to make his delivery while Mister George was visiting. He knows the route mister George took.

The city is modeled with intersections and two-way streets connecting them. For each street, Luka knows how much time he needs to traverse it (mister George needs the same amount of time).

For example, if Mister George starts traversing a street during minute 10 and needs 5 minutes to exit it, this street will be blocked during minutes 10, 11, 12, 13 and 14. Luka can enter the street during minutes 9 and earlier, or 15 and later.

Write a program that calculates the least amount of time Luka needs to make his delivery, if he starts driving K minutes after the arrival of Mister George.

Input

The first line contains two integers N and M ($2 \leq N \leq 1000$, $2 \leq M \leq 10\,000$), the number of intersections and the number of streets. The intersections are numbered 1 to N.

The second line contains four integers A, B, K and G ($1 \leq A, B \leq N$, $0 \leq K \leq 1000$, $0 \leq G \leq 1000$). These are, in order:

- The intersection where Luka starts;
- The intersection Luka must get to;
- The difference in starting times between mister George and Luka (Luka starts at intersection A exactly K minutes after mister George starts his route);
- The number of intersections on Mister George's route.

The third line contains G integers, the labels of intersections mister George will visit. Every pair of adjacent integers denotes a street he will traverse. That street will exist and Mister George will traverse every street at most once.

Each of the following M lines contains three integers A, B and L, meaning that there is a street between intersection A and B, and it takes L minutes to traverse. L will be between 1 and 1000.

Output

Output the least amount of time (in minutes) Luka needs to make his delivery.

Sample test data

input

```
6 5
1 6 20 4
5 3 2 4
1 2 2
2 3 8
2 4 3
3 6 10
3 5 15
```

output

```
21
```

input

```
8 9
1 5 5 5
1 2 3 4 5
1 2 8
2 7 4
2 3 10
6 7 40
3 6 5
6 8 3
4 8 4
4 5 5
3 4 23
```

output

```
40
```

5. PRINCEZA

Luka parked his truck near the lake. The lake is inhabited by the frog Barica, who jumps across N plants floating on the lake's surface. Knowing a fair number of folk tales, Luka knows that if he kisses Barica, she will turn into a beautiful princess. However, he needs to catch her first!

Assuming an aerial view, the position of a plant on the lake's surface can be defined with a pair of coordinates. From plant (x, y) Barica can jump:

- To plant (x+P, y+P), for any positive integer P. Call this direction A.
- To plant (x+P, y−P), for any positive integer P. Call this direction B.
- To plant (x−P, y+P), for any positive integer P. Call this direction C.
- To plant (x−P, y−P), for any positive integer P. Call this direction D.

Barica selects one of the four directions and jumps onto the first plant in the chosen direction. If there is no plant in the selected direction, Barica stays where she is. After Barica jumps, **the plant she jumped from sinks** and disappears.

Knowing the locations of the plants and the sequence of directions Barica chooses, Luka wants to determine coordinates of the plant Barica will end up on. Luka will wait for her at that plant, ambush her and kiss her.

Write a program that solves Luka's problem and helps him turn Barica into a beautiful princess.

Input

The first line contains two integers N and K ($1 \leq N, K \leq 100\,000$), the number of plants and the number of attempted jump.

The second line contains K letters each of which is 'A', 'B', 'C' or 'D'. These letters represent in order the directions in which Barica attempts to jump.

Each of the following N lines contains two integers X and Y ($0 \leq X \leq 1\,000\,000\,000$, $0 \leq Y \leq 1\,000\,000\,000$), the coordinates of one plant. Barica is initially located on the first plant.

Output

Output Barica's final coordinates.

Sample test data

input

```
7 5
ACDBB
5 6
8 9
4 13
1 10
7 4
10 9
3 7
```

output

```
7 4
```

input

```
6 12
AAAAAABCCCDD
1 1
2 2
3 3
4 4
5 3
6 2
```

output

```
5 3
```

6. CESTARINE

In a single day, N of Luka's trucks travel a specific highway. The highway has a number of exits and entrances. An exit with a particular number is in the same location as the entrance with that number.

Upon entering the highway, a truck driver receives a ticket which indicates the entrance he used. When exiting, the driver pays a toll equal to the absolute difference of the entrance and exit numbers. For example, if a ticket says he used entrance 30, then exiting at exit 12 will cost him 18.

Luka has figured out a way to save toll money that his company daily spends. Any two drivers can meet on the highway and exchange tickets, even if their routes don't overlap. Tickets can be exchanged an arbitrary number of times.

However, **a driver cannot use an exit if his ticket says he used the same entrance**, since that would be suspicious.

Write a program that calculates the least total amount of tolls that the drivers can achieve by exchanging tickets.

Input

The first line contains the integer N ($1 \leq N \leq 100000$), the number of trucks.

Each of the following N lines contains two distinct integers between 1 and 1000000. These are in order the entrance and exit numbers of one truck.

No two trucks will use the same highway entrance or the same exit.

Output

Output the least total amount of tolls Luka's company must pay.

Note: use 64-bit integer types (long long in C/C++, int64 in Pascal).

Sample test data

input

```
3
3 65
45 10
60 25
```

output

```
32
```

input

```
3
5 5
6 7
8 8
```

output

```
5
```

In the first example, the first and third drivers will exchange tickets. After this, the second and third drivers exchange tickets. After this, the drivers will have the tickets 60, 3, 45, respectively. The total amount in tolls is $|65-60| + |10-3| + |25-45| = 32$.