Problem A
Being Solarly Systematic

Professor Braino Mars is one of the top researchers in the field of solar system creation. He runs various simulations to test out his theories on planet formation, but he's old school and all of these simulations are done by hand. It's time for Braino to enter the 21^{st} century, and he's asked you to help automate his simulations.

One of Prof. Mars' simulations models how small planetoids collide over time to form larger planets. To model this process he divides the space which the planetoids inhabit into an $n_x \times n_y \times n_z$ grid of cubes, where each cube can hold at most one planetoid. Each planetoid has an initial mass m, an initial location (x, y, z) in the grid and a velocity (v_x, v_y, v_z) indicating the number of cubes per second the planetoid travels through in each dimension. For example, if a planetoid is initially in location $(1, 3, 2)$ and has velocity $(3, -1, 2)$, then after 1 second it will be in location $(4, 2, 4)$, after 2 seconds it will be in location $(7, 1, 6)$, and so on. The planetoid paths wrap around in all dimensions, so if, for example, the planetoid described above resides in an $8 \times 8 \times 8$ space, its next two locations will be $(2, 0, 0)$ and $(5, 7, 2)$ (note that all cube indices start at 0). When two or more planetoids collide, they form one larger planetoid which has a mass equal to the sum of the colliding planetoids' masses and a velocity equal to the average of the colliding velocities, truncating to the nearest integer. So if a planetoid of mass 12 with velocity $(5, 3, -2)$ collides with another planetoid of mass 10 and velocity $(8, -6, 1)$ the resulting planetoid has mass 22 and velocity $(6, -1, 0)$ (these values correspond to the first sample input.) For simplicity, Prof. Mars only considers collisions that happen at integer time steps, and when no more collisions are possible, the planetoids are then considered full-fledged planets.

Given an initial set of planetoids, Prof. Mars is interested in determining how many planets will form and what their orbits are. Armed with your implementation of his model, he should now be able to answer these questions much more easily.

Input

The input will start with a line containing four positive integers $n\ n_x\ n_y\ n_z$, where $n \leq 100$ is the number of planetoids, and n_x, n_y and n_z are the dimensions of the space the planetoids reside in, where $n_x, n_y, n_z \leq 1000$.

After this are n lines of the form $m\ x\ y\ z\ v_x\ v_y\ v_z$, specifying the mass, initial location and initial velocity of each planetoid at time $t = 0$, where $1 \leq m \leq 100, 0 \leq x < n_x, 0 \leq y < n_y, 0 \leq z < n_z$, and $-1000 \leq v_x, v_y, v_z \leq 1000$. No two planetoids will start in the same initial location.

Output

Output an integer p indicating the number of planets in the system after no more collisions can occur. After this output p lines, one per planet, listing a planet identifier Pi, $(0 \leq i < p)$, the mass, location and velocity of each planet. Use the location of the planets at the time that the last collision occurred.

If no collisions occur, then use their location at time $t = 0$.

The planets should be ordered from largest mass to smallest; break ties by using the lexicographic ordering of the x, y, z location of the planet, starting with the smallest x value.

Sample Input 1

```
2 8 8 8
12 4 1 4 5 3 -2
10 1 2 1 8 -6 1
```

Sample Output 1

```
1
P0: 22 1 4 2 6 -1 0
```

Sample Input 2

```
2 10 20 30
10 1 0 0 2 0 0
15 2 0 0 4 0 0
```

Sample Output 2

```
2
P0: 15 2 0 0 4 0 0
P1: 10 1 0 0 2 0 0
```

Problem B
Delete This!

Well, it's time. Andrew has been accumulating file after file on his computer and could never bring himself to delete any single one of them ("You know Algol might make a comeback, so I better not delete any of those files" is one of a large number of his justifications). But he realizes that not only is it wasting a lot of disk space, but it's making it hard to find anything when he displays his files on the screen as icons.

Because of the sheer number of files that must be gotten rid of, Andrew would like to use as few delete operations as possible. He can delete multiple files at one time if their icons are all clustered together in a rectangular area on his screen by using the mouse to outline a box around them and then hitting delete (an icon is considered in the box if its center is in the box). This also requires that there are no icons in the box of files that he wants to keep. He figures that maybe if he moves file icons around, he can easily put all of the icons into such a rectangular area, perhaps moving some icons out as well.

For example, in the figure below there are three files to delete (black icons) and two to keep (white icons). By moving two of them as shown in the figure on the right, all of the three icons of files to be deleted can be grouped together for one delete operation (note that there are many other ways to move two icons to accomplish this, but no way to do it by just moving one).

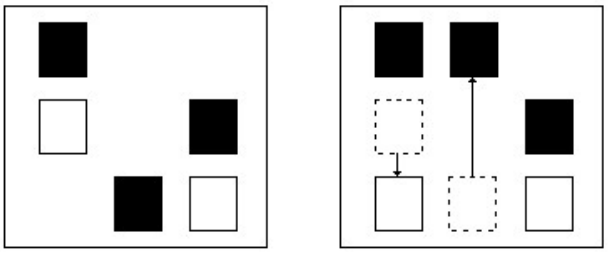


Figure B.1

Since he must clean out every directory in his file system, he would like to know the following: given a layout of file icons on the screen, what is the minimum number of icons to move so that he can delete all of the appropriate files with one delete command?

Input

The input will start with four integers n_r n_c n m which indicate the number of pixel rows and columns in the screen $(1 \leq n_r, n_c \leq 10000)$, the number of file icons on the screen to be deleted (n) and the number of file icons on the screen that should not be deleted (m), where $n + m \leq 100$. After this will be a set of $2(n + m)$ integers indicating the location of the $n + m$ files, the first n of which are the files to be deleted. Each pair of numbers r c will specify the row and col of the upper left corner of the file icon, where $0 \leq r < n_r$ and $0 \leq c < n_c$. All icons are 15 pixels high by 9 pixels wide in size and no two icons will be at the same location, though they may overlap, and at least one pixel of the icon must always reside on the screen (both initially and after they've been moved). Edges of a delete rectangle lie on pixel boundaries.

Output

Output the minimum number of file icons that must be moved in order to delete all the appropriate files in one delete operation.

Sample Input 1

```
80 50 3 2
75 5 25 20 50 35
50 5 25 35
```

Sample Output 1

```
2
```

Sample Input 2

```
100 100 1 1
50 50 80 80
```

Sample Output 2

```
0
```

Problem C
Transportation Delegation

You have just been hired by Amalgamated, Inc. in the country of Acmania to oversee the transportation of raw materials to the company's factories. Each supplier of raw materials and each factory resides in one of Acmania's states. No state has both a factory and a supplier (and never more than one of either) and there are arcane laws governing which transportation companies can transport materials across state lines. Because of the fierce competition between factories and between suppliers each transportation company handles the output of at most one raw material site and delivers to at most one factory (or to another transportation company). Each supplier can produce enough material to contract with at most one factory and no factory will contract with more than one supplier. Your job is to determine the maximum number of factories that can be supplied with raw materials.

For example, suppose that there are three suppliers in states A, B and C, and three factories in states D, E and F. Let's say you contract three transportation firms: firm 1 can transport between states A, E and G; firm 2 can transport between states A, C and E; and firm 3 can transport between states B, D and F. In this case, you can supply at most two factories (for example, factory E can be supplied from supplier A using firm 1, and factory F can be supplied from supplier B using firm 3). If you find a fourth firm that transports between states G and F then you can supply all three factories: factory D can be supplied from B using firm 3, factory E can be supplied from C using firm 2, and factory F can be supplied from A using firms 1 and 4.

Input

The input will start with four positive integers $s\ r\ f\ t$ indicating the number of states, raw material sites, factories and transportation companies, where $1 \leq r, f \leq 200$, $r + f \leq s \leq 600$ and $1 \leq t \leq 1000$.

Next will follow a line containing r state names, one for each raw material site.

The next line will contain f state names, one for each factory site.

Finally there will be t lines, one for each transportation company. Each of these lines will start with an integer n, $1 \leq n \leq s$, indicating the number of states the company is allowed to work in, followed by n state names. No state will contain both a raw material site and a factory site.

All state names will be alphabetic strings with no blanks.

Output

Output the maximum number of factories that can be supplied with raw materials.

Sample Input 1	Sample Output 1
7 3 3 3 A B C D E F 3 A E G 3 A C E 3 B D F	2

Sample Input 2	Sample Output 2
7 3 3 4 A B C D E F 3 A E G 3 A C E 3 B D F 2 G F	3

Problem D
Rings

Dee Siduous is a botanist who specializes in trees. A lot of her research has to do with the formation of tree rings, and what they say about the growing conditions over the tree's lifetime. She has a certain theory and wants to run some simulations to see if it holds up to the evidence gathered in the field.

One thing that needs to be done is to determine the expected number of rings given the outline of a tree. Dee has decided to model a cross section of a tree on a two dimenional grid, with the interior of the tree represented by a closed polygon of grid squares. Given this set of squares, she assigns rings from the outer parts of the tree to the inner as follows: calling the non-tree grid squares "ring 0", each ring n is made up of all those grid squares that have at least one ring $(n-1)$ square as a neighbor (where neighboring squares are those that share an edge).

An example of this is shown in the figure below.

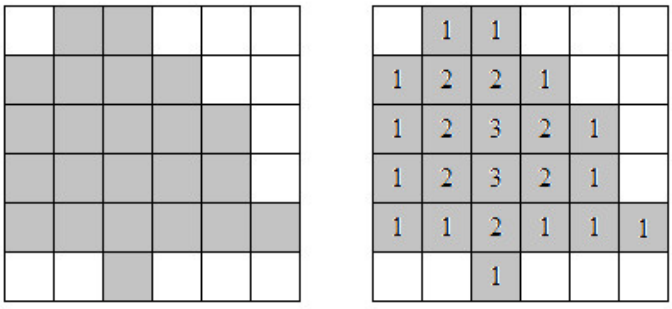


Figure D.1

Most of Dee's models have been drawn on graph paper, and she has come to you to write a program to do this automatically for her. This way she'll use less paper and save some . . . well, you know.

Input

The input will start with a line containing two positive integers n m specifying the number of rows and columns in the tree grid, where $n, m \leq 100$. After this will be n rows containing m characters each. These characters will be either 'T' indicating a tree grid square, or '.'.

Output

Output a grid with the ring numbers. If the number of rings is less than 10, use two characters for each grid square; otherwise use three characters for each grid square. Right justify all ring numbers in the grid squares, and use '.' to fill in the remaining characters.

If a row or column does not contain a ring number it should still be output, filled entirely with '.'s.

Sample Input 1

```
6 6
.TT...
TTTT..
TTTTT.
TTTTT.
TTTTTT
..T...
```

Sample Output 1

```
...1.1......
.1.2.2.1....
.1.2.3.2.1..
.1.2.3.2.1..
.1.1.2.1.1.1
.....1......
```

Sample Input 2

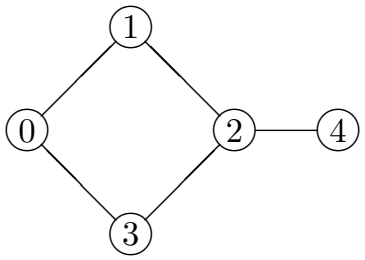
```
3 4
TT..
TT..
....
```

Sample Output 2

```
.1.1....
.1.1....
........
```

Problem E
Squawk Virus

Oh no! Hackers are threatening to shut down Twitface, the premier social networking site. By taking advantage of lax security protocols, nefarious cyber-bandits have developed a virus that spreads from user to user, amplifying over time and eventually bringing the network to its knees from massive congestion. Normally users have to manually send messages to one another (squawking), but these ne'er-do-wells have figured out how to circumvent that rule, and have created squawks that spawn more squawks without user intervention. In particular, any time a user gets an infected squawk, one minute later it broadcasts an infected squawk to all its neighbors in the network (for purposes of this problem we assume that each neighbor gets the squawk exactly 1 minute after the initial user is infected). If a user receives multiple squawks at any point, the next minute it broadcasts that many squawks to all of its neighbors. For example, consider the following network:



If user 0 is infected at time $t = 0$, then at time $t = 1$ users 1 and 3 get 1 squawk each, at time $t = 2$ users 0 and 2 get 2 squawks each, and at time $t = 3$, users 1 and 3 get 4 squawks each and user 4 gets 2 squawks.

Given the layout of a social network and an initial infection site, you need to determine how many squawks are made at some given time t. In the above example the number of squawks would be 2, 4 and 10 at times 1, 2 and 3, respectively.

Input

The input will start with a line containing 4 integers $n\ m\ s\ t$ indicating the number of users $(1 \leq n \leq 100)$, the number of links between users $(0 \leq m \leq n(n-1)/2)$, the index of the initially infected user $(s < n)$, and the number of minutes $(t < 10)$. Next will follow m lines, each consisting of two integers $x\ y$, $(0 \leq x, y < n)$ indicating that users x and y are connected. Connections are symmetric and no two connections will be the same.

Output

Output the number of squawks sent at the specified time t.

Sample Input 1	Sample Output 1
4 3 1 4 0 1 1 2 2 3	8

Sample Input 2	Sample Output 2
5 5 0 3 0 1 0 3 1 2 2 3 2 4	10

Problem F: ASCII Addition

Nowadays, there are smartphone applications that instantly translate text and even solve math problems if you just point your phone's camera at them. Your job is to implement a much simpler functionality reminiscent of the past – add two integers written down as ASCII art.

An *ASCII art* is a matrix of characters, exactly 7 rows high, with each individual character either a dot or the lowercase letter x.

An expression of the form $a + b$ is given, where both a and b are positive integers. The expression is converted into ASCII art by writing all the expression characters (the digits of a and b as well as the + sign) as 7×5 matrices, and concatenating the matrices together with a single column of dot characters between consecutive individual matrices. The exact matrices corresponding to the digits and the + sign are as folows:

```
xxxxx  ....x  xxxxx  xxxxx  x...x  xxxxx  xxxxx  xxxxx  xxxxx  xxxxx  .....
x...x  ....x  ....x  ....x  x...x  x....  x....  ....x  x...x  x...x  ..x..
x...x  ....x  ....x  ....x  x...x  x....  x....  ....x  x...x  x...x  ..x..
x...x  ....x  xxxxx  xxxxx  xxxxx  xxxxx  xxxxx  ....x  xxxxx  xxxxx  xxxxx
x...x  ....x  x....  ....x  ....x  ....x  x...x  ....x  x...x  ....x  ..x..
x...x  ....x  x....  ....x  ....x  ....x  x...x  ....x  x...x  ....x  ..x..
xxxxx  ....x  xxxxx  xxxxx  ....x  xxxxx  xxxxx  ....x  xxxxx  xxxxx  .....
```

Given an ASCII art for an expression of the form $a + b$, find the result of the addition and write it out in the ASCII art form.

Input

Input consists of exactly 7 lines and contains the ASCII art for an expression of the form $a + b$, where both a and b are positive integers consisting of at most 9 decimal digits and written without leading zeros.

Output

Output 7 lines containing ASCII art corresponding to the result of the addition, without leading zeros.

Example

input

```
....x.xxxxx.xxxxx.x...x.xxxxx.xxxxx.xxxxx.......xxxxx.xxxxx.xxxxx
....x.....x.....x.x...x.x.....x.........x...x...x...x.x...x.x...x
....x.....x.....x.x...x.x.....x.........x...x...x...x.x...x.x...x
....x.xxxxx.xxxxx.xxxxx.xxxxx.xxxxx.....x.xxxxx.xxxxx.xxxxx.x...x
....x.x.........x.....x.....x.x...x.....x...x...x...x.....x.x...x
....x.x.........x.....x.....x.x...x.....x...x...x...x.....x.x...x
....x.xxxxx.xxxxx.....x.xxxxx.xxxxx.....x.......xxxxx.xxxxx.xxxxx
```

output

```
....x.xxxxx.xxxxx.xxxxx.x...x.xxxxx.xxxxx
....x.....x.....x.x.....x...x.x.........x
....x.....x.....x.x.....x...x.x.........x
....x.xxxxx.xxxxx.xxxxx.xxxxx.xxxxx.....x
....x.x.........x.....x.....x.....x.....x
....x.x.........x.....x.....x.....x.....x
....x.xxxxx.xxxxx.xxxxx.....x.xxxxx.....x
```

Problem G
Tray Bien

André Claude Marzipan is the head chef at the French restaurant Le Chaud Chien. He owns a vast number of baking trays, which come in two sizes: 1 foot by 1 foot, and 1 foot by 2 feet. He stores them along 3-foot deep shelves of various lengths. For example, on a shelf that is 5 feet long, he might store baking trays in either of the two ways shown below:

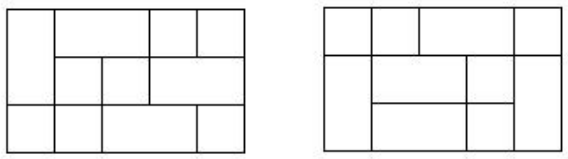


Figure G.1

Of course, there are many more than just these two ways, and in his off hours André often wonders how many different ways he can place trays on a given shelf. André is a bit of *un maniaque du rangement* (neat freak), so he insists that the trays are always aligned along the two axes defined by the shelf edges, that the edges of the trays are always 1 foot multiples away from any edge, and that no portion of a tray extends beyond the shelf. The matter is complicated by the fact that often there are locations on the shelf where he does not want to put any baking trays, due to leaks above the shelf, dents in the shelf's surface, etc. Since André is more adept at cuisine than counting, he needs a little help.

Input

The input consists of two lines: the first line will contain two integers m n, where $1 \leq m \leq 24$ indicates the length of the shelf (which is always 3-feet deep) and n indicates the number of bad locations on the shelf. The next line will contain n coordinate pairs x y indicating the locations where trays should not be placed, where $0 < x < m$ and $0 < y < 3$. No location will have integer coordinates and coordinates are specified to the nearest hundredth. If $n = 0$, this second line will be blank.

Output

Output the number of ways that trays could be placed on the shelf.

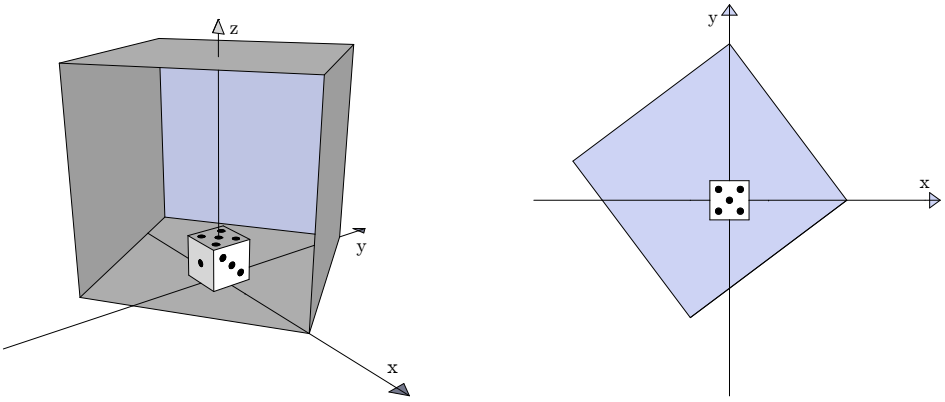
Sample Input 1	Sample Output 1
4 0	823

Sample Input 2	Sample Output 2
4 2 0.29 2.44 2.73 1.8	149

Problem H: Hovering Hornet

You have managed to trap a hornet inside a box lying on the top of your dining table. Unfortunately, your playing dice is also trapped inside – you cannot retrieve it and continue your game of Monopoly without risking the hornet's wrath. Instead, you pass your time calculating the expected number of spots on the dice visible to the hornet.

The hornet, the dice and the box are located in the standard three-dimensional coordinate system with the x coordinate growing eastwards, the y coordinate growing northwards and the z coordinate growing upwards. The surface of the table corresponds to the x-y plane.



Perspective and the birds-eye view of the second example input

The dice is a $1 \times 1 \times 1$ cube, resting on the table with the center of the bottom side exactly in the origin. Hence, the coordinates of its two opposite corners are $(-0.5,\ -0.5,\ 0)$ and $(0.5,\ 0.5,\ 1)$. The top side of the dice has 5 spots, the south side 1 spot, the east side 3 spots, the north side 6 spots, the west side 4 spots and the (invisible and irrelevant) bottom side 2 spots.

The box is a $5 \times 5 \times 5$ cube also resting on the table with the dice in its interior. The box is specified by giving the coordinates of its bottom side – a 5×5 square.

Assume the hornet is hovering at a uniformly random point in the (continuous) space inside the box not occupied by the dice. Calculate the expected number of spots visible by the hornet. The dice is opaque and, hence, the hornet sees a spot only if the segment connecting the center of the spot and the location of the hornet does not intersect the interior of the dice.

Input

Input consists of 4 lines. The k-th line contains two floating-point numbers x_k and y_k ($-5 \leq x_k,\ y_k \leq 5$) – coordinates of the k-th corner of the bottom side of the box in the x-y plane. The coordinates are given in the counterclockwise direction and they describe a square with the side length of exactly 5.

The box fully contains the dice. The surfaces of the box and the dice do not intersect or touch except along the bottom sides.

Output

Output a single floating point number – the expected number of spots visible. The solution will be accepted if the absolute or the relative difference from the judges solution is less than 10^{-6}.

Example

input

```
-2.5 -1.5
2.5 -1.5
2.5 3.5
-2.5 3.5
```

output

```
10.6854838710
```

input

```
3 0
0 4
-4 1
-1 -3
```

output

```
10.1226478495
```

Problem I: Ice Igloos

A fishing village built on the surface of a frozen lake far north in the arctic is endangered by global warming – fractures are starting to form on the lake surface. The village consists of n igloos of spherical shape, each occupying a circular area of the surface.

An igloo can be represented as a circle in the coordinate plane: the center of the circle is a point with integer coordinates, while the radius is a positive floating-point number less than 1 with exactly one fractional digit.

Given the locations of possible ice fractures, the villagers would like to know how many igloos are affected by each. Formally, given q queries where each query is a straight line segment defined by the two endpoints, find the number of igloos each segment intersects. A segment intersects an igloo if it has at least one point in common with the interior of the circle.

Input

The first line contains an integer n ($1 \leq n \leq 100\,000$) - the number of igloos. Each of the following n lines contains three numbers x, y and r – the coordinates of the center and the radius of one igloo. The coordinates x and y are integers such that $1 \leq x, y \leq 500$, while r is a floating-point number with exactly one fractional digit such that $0 < r < 1$. No two igloos will intersect or touch.

The following line contains an integer q ($1 \leq q \leq 100\,000$) - the number of queries. Each of the following q lines contains four integers x_1, y_1, x_2, y_2 ($1 \leq x_1, y_1, x_2, y_2 \leq 500$) - the coordinates of the two endpoints of the segment. The two endpoints will be different. Endpoints may be inside igloos.

You may assume that, for every igloo i and the segment s, the square of the distance between s and the center of i is either less than $r^2 - 10^{-5}$ or greater than $r^2 + 10^{-5}$ where r is the radius of the igloo i.

Output

Output should consist of q lines. The k-th line should contain a single integer – the number of igloos that are intersected by the k-th segment.

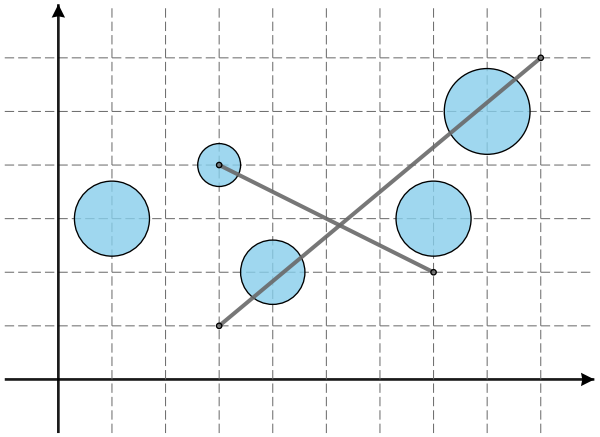
Example

input

```
5
4 2 0.6
7 3 0.7
8 5 0.8
1 3 0.7
3 4 0.4
2
3 1 9 6
3 4 7 2
```

output

```
2
1
```



Problem J: Juice Junctions

You have been hired to upgrade an orange juice transport system in an old fruit processing plant. The system consists of pipes and junctions. All the pipes are bidirectional and have the same flow capacity of 1 liter per second. Pipes may join at junctions and each junction joins *at most three pipes*. The flow capacity of the junction itself is unlimited. Junctions are denoted with integers from 1 to n.

Before proposing the upgrades, you need to analyze the existing system. For two different junctions s and t, the s-t flow is defined as the maximum amount of juice (in liters per second) that could flow through the system if the source was installed at junction s and the sink at junction t. For example, in the system from the first example input below, the 1-6 flow is 3 while the 1-2 flow is 2.

Find the sum of all a-b flows for every pair of junctions a and b such that $a < b$.

Input

The first line contains two integers n and m ($2 \leq n \leq 3\,000$, $0 \leq m \leq 4\,500$) – the number of junctions and the number of pipes. Each of the following m lines contains two different integers a and b ($1 \leq a, b \leq n$) which describe a pipe connecting junctions a and b.

Every junction will be connected with at most three other junctions. Every pair of junctions will be connected with at most one pipe.

Output

Output a single integer – the sum of all a-b flows for every pair of junctions a and b such that $a < b$.

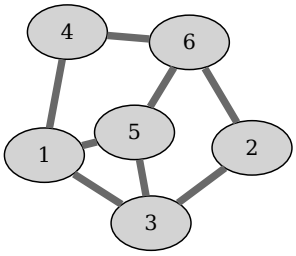
Example

input

```
6 8
1 3
2 3
4 1
5 6
2 6
5 1
6 4
5 3
```

output

```
36
```

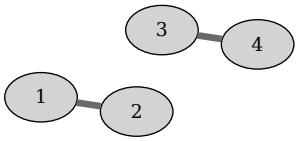


input

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

output

```
2
```



Problem K: Kernel Knights

Jousting is a medieval contest that involves people on horseback trying to strike each other with wooden lances while riding at high speed. A total of $2n$ knights have entered a jousting tournament – n knights from each of the two great rival houses. Upon arrival, each knight has challenged a single knight from the other house to a duel.

A *kernel* is defined as some subset S of knights with the following two properties:

- No knight in S was challenged by another knight in S.
- Every knight not in S was challenged by some knight in S.

Given the set of the challenges issued, find one kernel. It is guaranteed that a kernel always exists.

Input

The first line contains an integer n ($1 \leq n \leq 100\,000$) – the number of knights of each house. The knights from the first house are denoted with integers 1 through n, knights from the second house with integers $n+1$ through $2n$.

The following line contains integers $f_1, f_2, \ldots, f_n$ – the k-th integer f_k is the index of the knight challenged by knight k ($n+1 \leq f_k \leq 2n$).

The following line contains integers $s_1, s_2, \ldots, s_n$ – the k-th integer s_k is the index of the knight challenged by knight $n+k$ ($1 \leq s_k \leq n$).

Output

Output the indices of the knights in the kernel on a single line. If there is more than one solution, you may output any one.

Example

input

```
4
5 6 7 7
1 3 2 3
```

output

```
1 2 4 8
```

Problem L: Digit Division

We are given a sequence of n decimal digits. The sequence needs to be partitioned into one or more contiguous subsequences such that each subsequence, when interpreted as a decimal number, is divisible by a given integer m.

Find the number of different such partitions modulo $10^9 + 7$. When determining if two partitions are different, we only consider the locations of subsequence boundaries rather than the digits themselves, e.g. partitions 2|22 and 22|2 are considered different.

Input

The first line contains two integers n and m ($1 \leq n \leq 300\,000$, $1 \leq m \leq 1\,000\,000$) – the length of the sequence and the divisor respectively. The second line contains a string consisting of exactly n digits.

Output

Output a single integer – the number of different partitions modulo $10^9 + 7$.

Example

input	output
4 2 1246	4

input	output
4 7 2015	0

Problem M: Looping Labyrinth

A labyrinth is obtained by tiling the entire plane with a *pattern* – a rectangular grid consisting of n rows and m columns where every cell is either empty or blocked. The result is an infinite grid of cells with the pattern repeating in all four directions.

Formally, suppose we denote both rows and columns of the infinite grid with integers (including the negative integers). The row number increases as we move downwards in the grid, while the column number increases as we go to the right. The cell at coordinates $(0, 0)$ is called the *origin*. The labyrinth is obtained by copying the pattern (without mirroring or rotation) to every n-by-m rectangular area that, in the upper-left corner, has a cell with the row number divisible by n and the column number divisible by m. In particular, the upper-left corner of the pattern gets copied to the origin, while the lower-right corner gets copied to the cell with coordinates $(n - 1, m - 1)$.

To escape the labyrinth starting from a particular cell, we need to reach the origin via a sequence of empty cells, going up, down, left or right in each step.

You are given a pattern and a sequence of possible starting cells. For each starting cell determine if it is possible to escape the labyrinth.

Input

The first line contains two integers n and m ($1 \leq n,\ m \leq 100$) – the number of rows and columns in the pattern, respectively. Each of the following n lines contains a string of exactly m characters describing one row of the pattern. The character `#` denotes a blocked cell while the dot character denotes an empty cell.

The following line contains an integer q ($1 \leq q \leq 200,000$) – the number of starting cells. The k-th of the following q lines contains two integers r and c ($-10^9 \leq r,\ c \leq 10^9$) – the row and column of the k-th starting cell.

The origin and all starting cells will be empty.

Output

Output should consist of q lines. The k-th line should contain the word `yes` if it is possible to exit the labyrinth from the k-th starting cell and the word `no` otherwise.

Example

input

```
6 9
..#####..
..#...#..
......#..
..#####..
..#......
..#...#..
5
1 4
5 4
1 -5
5 -5
-1000000000 0
```

output

```
yes
no
no
yes
yes
```