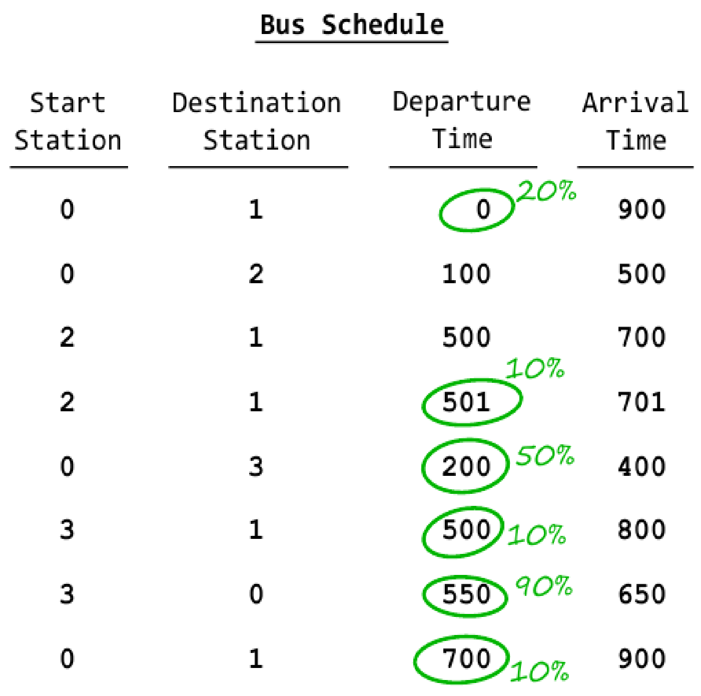
Problem A. Catch the Plane

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	1024 megabytes

Your plane to the ICPC Finals departs in a short time, and the only way to get to the airport is by bus. Unfortunately, some of the bus drivers are considering going on strike, so you do not know whether you can get to the airport on time. Your goal is to plan your journey in such a way as to maximize the probability of catching your plane.

You have a detailed map of the city, which includes all the bus stations. You are at station 0 and the airport is at station 1. You also have a complete schedule of when each bus leaves its start station and arrives at its destination station. Additionally, for each bus you know the probability that it is actually going to run as scheduled, as opposed to its driver going on strike and taking the bus out of service. Assume all these events are independent. That is, the probability of a given bus running as planned does not change if you know whether any of the other buses run as planned.

If you arrive *before* the departure time of a bus, you can transfer to that bus. But if you arrive exactly at the departure time, you will not have enough time to get on the bus. You cannot verify ahead of time whether a given bus will run as planned – you will find out only when you try to get on the bus. So if two or more buses leave a station at the same time, you can try to get on only one of them.



Bus Schedule

Start Station	Destination Station	Departure Time	Arrival Time
0	1	0 20%	900
0	2	100	500
2	1	500	700
2	1	501 10%	701
0	3	200 50%	400
3	1	500 10%	800
3	0	550 90%	650
0	1	700 10%	900

Figure A.1: Bus schedule corresponding to Sample Input 1.

Consider the bus schedule shown in Figure A.1. It lists the start and destination stations of several bus routes along with the departure and arrival times. You have written next to some of these the probability that that route will run. Bus routes with no probability written next to them have a 100% chance of running. You can try catching the first listed bus. If it runs, it will take you straight to the airport, and you can stop worrying. If it does not, things get more tricky. You could get on the second listed bus to station 2. It is certain to leave, but you would be too late to catch the third listed bus which otherwise would have delivered you to the airport on time. The fourth listed bus – which you can catch – has only a 0.1 probability of actually running. That is a bad bet, so it is better to stay at station 0 and wait for the fifth listed bus. If you catch it, you can try to get onto the sixth listed bus to the airport; if that does not run, you still have the chance of returning to station 0 and catching the last listed bus straight to the airport.

Input

The first line of input contains two integers m ($1 \le m \le 10^6$) and n ($2 \le n \le 10^6$), denoting the number of buses and the number of stations in the city. The next line contains one integer k ($1 \le k \le 10^{18}$), denoting the time by which you must arrive at the airport.

Each of the next m lines describes one bus. Each line contains integers a and b ($0 \le a, b < n$, $a \neq b$), denoting the start and destination stations for the bus. Next are integers s and t ($0 \le s < t \le k$), giving the departure time from station a and the arrival time at station b. The last value on the line is p ($0 \le p \le 1$, with at most 10 digits after the decimal point), which denotes the probability that the bus will run as planned.

Output

Display the probability that you will catch your plane, assuming you follow an optimal course of action. Your answer must be correct to within an absolute error of 10^{-6}.

Examples

standard input	standard output

8 4
1000
0 1 0 900 0.2
0 2 100 500 1.0
2 1 500 700 1.0
2 1 501 701 0.1
0 3 200 400 0.5
3 1 500 800 0.1
3 0 550 650 0.9
0 1 700 900 0.1
	0.3124

4 2
2
0 1 0 1 0.5
0 1 0 1 0.5
0 1 1 2 0.4
0 1 1 2 0.2
	0.7

Problem B. Comma Sprinkler

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 seconds
Memory limit:	256 megabytes

Photo by Tanya Hart. Yarn pattern by Morgen Dämmerung.

As practice will tell you, the English rules for comma placement are complex, frustrating, and often ambiguous. Many people, even the English, will, in practice, ignore them, and, apply custom rules, or, no rules, at all.

Doctor Comma Sprinkler solved this issue by developing a set of rules that sprinkles commas in a sentence with no ambiguity and little simplicity. In this problem you will help Dr. Sprinkler by producing an algorithm to automatically apply her rules.

Dr. Sprinkler's rules for adding commas to an existing piece of text are as follows:

1. If a word anywhere in the text is preceded by a comma, find all occurrences of that word in the text, and put a comma before each of those occurrences, except in the case where such an occurrence is the first word of a sentence or already preceded by a comma.
2. If a word anywhere in the text is succeeded by a comma, find all occurrences of that word intext, and put a comma after each of those occurrences, except in the case where such an occurrence is the last word of a sentence or already succeeded by a comma.
3. Apply rules 1 and 2 repeatedly until no new commas can be added using either of them.

As an example, consider the text

```
please sit spot. sit spot, sit. spot here now here.
```

Because there is a comma after `spot` in the second sentence, a comma should be added after `spot` in the third sentence as well (but not the first sentence, since it is the last word of that sentence). Also, because there is a comma before the word `sit` in the second sentence, one should be added before that word in the first sentence (but no comma is added before the word `sit` beginning the second sentence because it is the first word of that sentence). Finally, notice that once a comma is added after `spot` in the third sentence, there exists a comma before the first occurrence of the word here. Therefore, a comma is also added before the other occurrence of the word here. There are no more commas to be added so the final result is

```
please, sit spot. sit spot, sit. spot, here now, here.
```

Input

The input contains one line of text, containing at least 2 characters and at most 1 000 000 characters. Each character is either a lowercase letter, a comma, a period, or a space. We define a word to be a maximal sequence of letters within the text. The text adheres to the following constraints:

- The text begins with a word.
- Between every two words in the text, there is either a single space, a comma followed by a space, or a period followed by a space (denoting the end of a sentence and the beginning of a new one).
- The last word of the text is followed by a period with no trailing space.

Output

Display the result after applying Dr. Sprinkler's algorithm to the original text.

Examples

standard input
`please sit spot. sit spot, sit. spot here now here.`
standard output
`please, sit spot. sit spot, sit. spot, here now, here.`

standard input
`one, two. one tree. four tree. four four. five four. six five.`
standard output
`one, two. one, tree. four, tree. four, four. five, four. six five.`

Problem C. Conquer the World

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 seconds
Memory limit:	256 megabytes

Bwahahahahaha!!! Your nemesis, the dashingly handsome spy Waco Powers, has at last fallen to your secret volcano base's deathtraps (or so you assume, being a little too busy to witness it firsthand). At long last, you are all set to CONQUER THE WORLD!

Nothing will stand in your way! Well, nothing except a minor problem of logistics. Your evil armies have announced that they will not continue carving their relentless path of destruction across the puny nations of the world without being paid. And unfortunately, you are running low on cash – a volcano lair has many wonderful qualities, but "reasonably affordable" is not one of them. You have had to pull funds from the travel budget to pay your ungrateful underlings. Now you are not sure how you will actually get your armies into position to CONQUER THE WORLD.

You have a map of the nations of the world and all your available transport routes between them. Each route connects two nations and has a fixed cost per army that uses it. The routes are laid out such that there is exactly one way to travel between any two nations. You know the current position of each of your armies and how many you will need to place permanently in each nation in order to subjugate it. How can you move the armies into place as cheaply as possible so you can CONQUER THE WORLD?

Input

The first line of input contains an integer n ($1 \le n \le 250\ 000$), the number of nations. This is followed by $n-1$ lines, each containing three integers u, v, and c ($1 \le u, v \le n$, $1 \le c \le 10^6$), indicating that there is a bidirectional route connecting nations u and v, which costs c per army to use.

Finally, another n lines follow, the i^{th} of which contains two non-negative integers x_i and y_i , indicating that there are currently x_i armies in nation i, and you need at least y_i armies to end up in that nation in the final configuration. The total number of armies (the sum of the x_i values) is at least the sum of the y_i values, and no more than 10^6.

Output

Display the minimum cost to move your armies such that there are at least y_i armies in nation i for all i.

Examples

standard input	standard output

3
1 2 5
3 1 5
2 1
5 0
1 3
	15

6
1 2 2
1 3 5
1 4 1
2 5 5
2 6 1
0 0
1 0
2 1
2 1
0 1
0 1
	9

Problem D. Gem Island

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Gem Island is a tiny island in the middle of the Pacific Ocean. Until recently, it was known as one of the poorest, but also most peaceful, places on Earth. Today, it is neither poor nor peaceful. What happened?

One sunny morning, not too long ago, all inhabitants of Gem Island woke up to a surprise. That morning, each of them suddenly held one sparkling gem in their hand. The gems had magically appeared overnight. This was cause for much rejoicing – everybody was suddenly rich, they could finally afford all the things they had ever dreamed of, and the name of their island made so much more sense now.

The next morning, one of the inhabitants woke up to another surprise – her gem had magically split into two gems! The same thing happened on each of the following nights, when exactly one of the gems (apparently uniformly at random among all the gems on the island) would split into two.

After a while, the inhabitants of Gem Island possessed a widely varying number of gems. Some hadlot and many had only a few. How come some inhabitants had more gems than others? Did they cheat, were they just lucky, or was something else at work?

The island elders have asked for your help. They want you to determine if the uneven distribution of gems is explained by pure chance. If so, that would greatly reduce tensions on the island.

The island has n inhabitants. You are to determine the gem distribution after d nights of gem splitting. In particular, you are interested in the expected number of gems collectively held by the r people with the largest numbers of gems. More formally, suppose that after d nights the numbers of gems held by the n inhabitants are listed in non-increasing order as $a_1 \geq a_2 \geq \ldots \geq a_n$. What is the expected value of $a_1 + \cdots + a_r$?

Input

The input consists of a single line containing the three integers n, d, and r ($1 \leq n$, $d \leq 500$, $1 \leq r \leq n$), as described in the problem statement above.

Output

Display the expected number of gems that the top r inhabitants hold after d nights, with an absolute or relative error of at most 10^{-6}.

Examples

standard input	standard output
`2 3 1`	`3.5`
`3 3 2`	`4.9`
`5 10 3`	`12.2567433`

Problem E. Getting a Jump on Crime

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Your friend Robin is a superhero. When you first found out about this, you figured "everybody needs a hobby, and this seems more exciting than stamp collecting", but now you are really thankful that somebody is doing something about the crime in your hometown.

Every night, Robin patrols the city by jumping from roof to roof and watching what goes on below. Naturally, superheroes need to respond to crises immediately, so Robin asked you for help in figuring out how to get around your hometown quickly.

Your hometown is built on a square grid, where each block is $w \times w$ meters. Each block is filled by a single building. The buildings may have different heights (see Figure E.1). To get from one building to another (not necessarily adjacent) building, Robin makes a single jump from the center of the roof of the first building to the center of the roof of the second building. Robin cannot change direction while in the

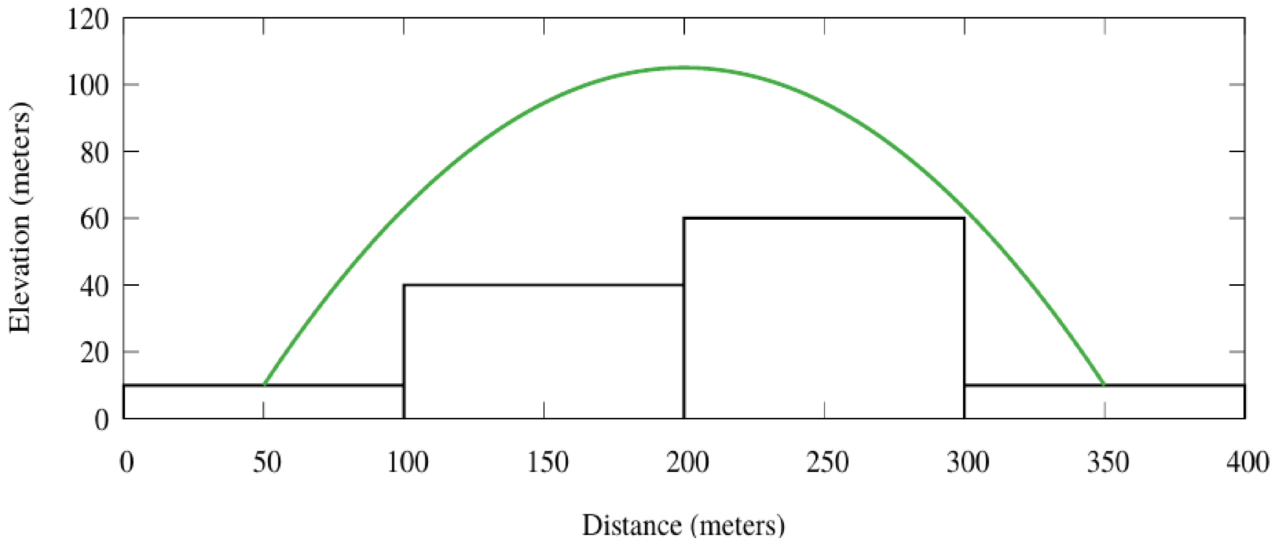


Figure E.1: Cross-section of buildings corresponding to the first sample input. Buildings are shown in black, and the jump from the roof at (1, 1) to the roof at (4, 1) is shown with a green line.

Of course, Robin only wants to perform jumps without colliding with any buildings. Such collisions do little damage to a superhero, but building owners tend to get irritated when someone crashes through their windows. You explain the physics to Robin: "All your jumps are done with the same initial velocity v, which has a horizontal component v_d towards the destination and vertical component v_h upwards,so $v_d^2 + v_h^2 = v^2$. As you travel, your horizontal velocity stays constant ($v_d(t) = v_d$), but your vertical velocity is affected by gravity ($v_h(t) = v_h - t \cdot g$), where $g = 9.80665 m/s^2$ in your hometown. Naturally, your cape allows you to ignore the effects of air resistance. This allows you to determine your flight path and ..." at which point you notice that Robin has nodded off – less math, more super-heroing!

So it falls to you: given a layout of the city and the location of Robin's secret hideout, you need to determine which building roofs Robin can reach, and the minimum number of jumps it takes to get to each roof.

Note that if Robin's jump passes over the corner of a building (where four buildings meet), then the jump needs to be higher than all four adjacent buildings.

Input

The input starts with a line containing six integers d_x, d_y, w, v, l_x, l_y . These represent the size $d_x \times d_y$ of the city grid ($1 \leq dx, dy \leq 20$) in blocks, the width of each building ($1 \leq w \leq 10^3$) in meters, Robin's takeoff velocity ($1 \leq v \leq 10^3$) in meters per second, and the coordinates (l_x, l_y) of Robin's secret hideout ($1 \leq l_x \leq d_x$, $1 \leq l_y \leq d_y$).

The first line is followed by a description of the heights of the buildings in the city grid. The description consists of d_y lines, each containing d_x non-negative integers. The j^{th} line contains the heights for buildings $(1, j)$, $(2, j), \ldots, (d_x, j)$. All heights are given in meters and are at most 10^3.

Output

Display the minimum number of jumps Robin needs to get from the secret hideout to the roof of each building. If there is no way to reach a building's roof, display `X` instead of the number of jumps. Display the buildings in the same order as given in the input file, split into d_y lines, each containing d_x values.

You may assume that changing the height of any building by up to 10^{-6} would not change the answers.

Examples

standard input	standard output

`4 1 100 55 1 1`
`10 40 60 10`
	`0 1 1 1`

`4 4 100 55 1 1`
`0 10 20 30`
`10 20 30 40`
`20 30 200 50`
`30 40 50 60`
	`0 1 1 2`
	`1 1 1 2`
	`1 1 X 2`
	`2 2 2 3`

Problem F. Go with the Flow

Input file:	`standard input`
Output file:	`standard output`
Time limit:	12 seconds
Memory limit:	256 megabytes

In typesetting, a "river" is a string of spaces formed by gaps between words that extends down several lines of text. For instance, Figure F.1 shows several examples of rivers highlighted in red (text is intentionally blurred to make the rivers more visible).

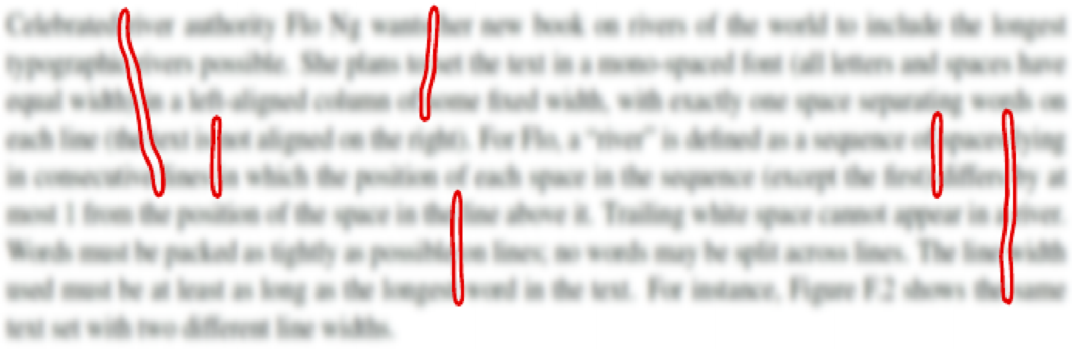


Figure F.1: Examples of rivers in typeset text.

Celebrated river authority Flo Ng wants her new book on rivers of the world to include the longest typographic rivers possible. She plans to set the text in a mono-spaced font (all letters and spaces have equal width) in a left-aligned column of some fixed width, with exactly one space separating words on each line (the text is not aligned on the right). For Flo, a "river" is defined as a sequence of spaces lying in consecutive lines in which the position of each space in the sequence (except the first) differs by at most 1 from the position of the space in the line above it. Trailing white space cannot appear in a river. Words must be packed as tightly as possible on lines; no words may be split across lines. The line width used must be at least as long as the longest word in the text. For instance, Figure F.2 shows the same text set with two different line widths.

Line width 14: River of length 4	**Line width 15: River of length 5**
`The Yangtze is`	`The Yangtze is`
`the third`	`the third`
`longest river`	`longest*river`
`in*Asia and`	`in Asia*and the`
`the*longest in`	`longest*in the`
`the*world to`	`world to*flow`
`flow*entirely`	`entirely*in one`
`in one country`	`country`

Figure F.2: Longest rivers (*) for two different line widths.

Given a text, you have been tasked with determining the line width that produces the longest river of spaces for that text.

Input

The first line of input contains an integer n ($2 \le n \le 2500$) specifying the number of words in the text. The following lines of input contain the words of text. Each word consists only of lowercase and uppercase letters, and words on the same line are separated by a single space. No word exceeds 80 characters.

Output

Display the line width for which the input text contains the longest possible river, followed by the length of the longest river. If more than one line width yields this maximum, display the shortest such line width.

Examples

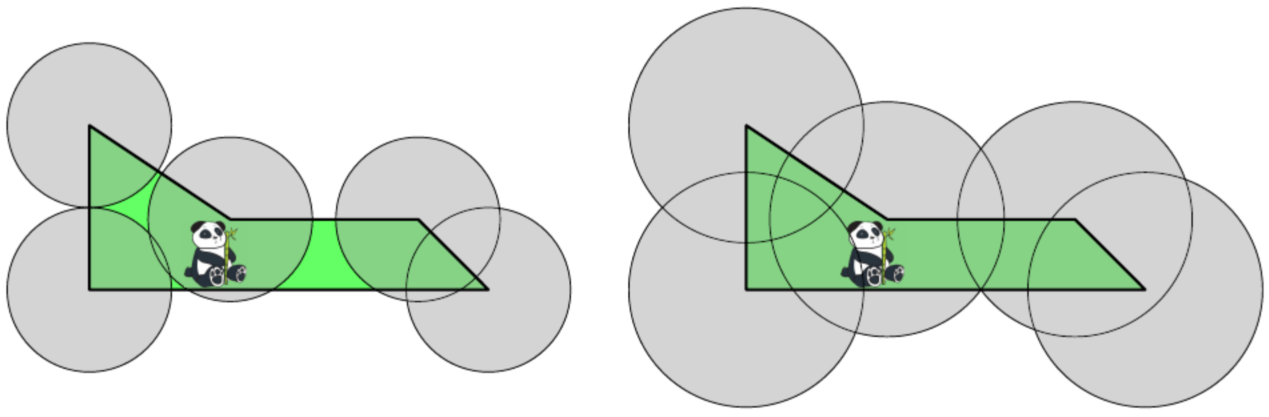
standard input	standard output
21 The Yangtze is the third longest river in Asia and the longest in the world to flow entirely in one country	15 5
25 When two or more rivers meet at a confluence other than the sea the resulting merged river takes the name of one of those rivers	21 6

Problem G. Panda Preserve

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	256 megabytes

Last month, Sichuan province secured funding to establish the Great Panda National Park, a natural preserve for a population of more than 1800 giant pandas. The park will be surrounded by a polygonal fence. In order for researchers to track the pandas, wireless receivers will be placed at each vertex of the enclosing polygon and each animal will be outfitted with a wireless transmitter. Each wireless receiver will cover a circular area centered at the location of the receiver, and all receivers will have the same range. Naturally, receivers with smaller range are cheaper, so your goal is to determine the smallest possible range that suffices to cover the entire park.

As an example, Figure G.1 shows the park described by the first sample input. Notice that a wireless range of 35 does not suffice (a), while the optimal range of 50 covers the entire park (b).



(a) An insufficient range for covering the park. (b) The minimal range for covering the park.

Figure G.1: Illustration of Sample Input 1.

Input

The first line of the input contains an integer n ($3 \leq n \leq 2000$) specifying the number of vertices of the polygon bounding the park. This is followed by n lines, each containing two integers x and y ($|x|$, $|y| \leq 10^4$) that give the coordinates (x, y) of the vertices of the polygon in counter-clockwise order. The polygon is simple; that is, its vertices are distinct and no two edges of the polygon intersect or touch, except that consecutive edges touch at their common vertex.

Output

Display the minimum wireless range that suffices to cover the park, with an absolute or relative error of at most 10^{-6}.

Examples

standard input	standard output
5	50
0 0	
170 0	
140 30	
60 30	
0 70	
5	51.538820320
0 0	
170 0	
140 30	
60 30	
0 100	
5	1.581138830
0 0	
1 2	
1 5	
0 2	
0 1	

Problem H. Single Cut of Failure

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 megabytes

The Intrusion and Crime Prevention Company (ICPC) builds intrusion detection systems for homes and businesses. The International Collegiate Programming Contest (in a strange coincidence also known as ICPC) is considering hiring the company to secure the room that contains the problem set for next year's World Finals.

The contest staff wants to prevent the intrusion attempts that were made in past years, such as rappelling down the outside of the building to enter through a window, crawling through air ducts, impersonating Bill Poucher, and the creative use of an attack submarine. For that reason, the problems will be stored in a room that has a single door and no other exits.

ICPC (the company) proposes to install sensors on the four sides of the door, where pairs of sensors are connected by wires. If somebody opens the door, any connected sensor pair will detect this and cause an alarm to sound.

The system has one design flaw, however. An intruder might cut the wires before opening the door. To assess the security of the system, you need to determine the minimum number of line segments that cut all wires. Figure H.1 shows two configurations of wires on the door (corresponding to the two sample inputs), and minimum-size cuts that intersect all wires.

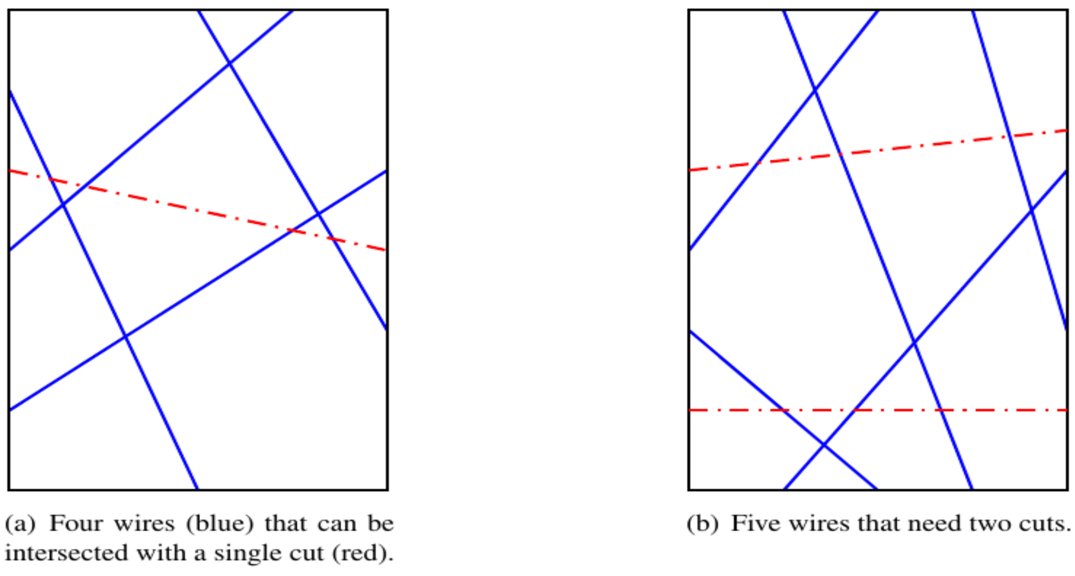

(a) Four wires (blue) that can be intersected with a single cut (red).
(b) Five wires that need two cuts.

Figure H.1: Illustrations of Sample Inputs 1 and 2.

Input

The input starts with a line containing three integers n, w, and h, which represent the number of wires installed ($1 \leq n \leq 10^6$) and the dimensions of the door ($1 \leq w, h \leq 10^8$). This is followed by n lines, each describing a wire placement. Each of these lines contains four integers x_1, y_1, x_2, and y_2 ($0 \leq x_1, x_2 \leq w$, $0 \leq y_1, y_2 \leq h$), meaning that a wire goes from (x_1, y_1) to (x_2, y_2). Each wire connects different sides of the door. No wire is anchored to any of the four corners of the door. All locations in the input are distinct.

Output

Display a minimum-size set of straight line cuts that intersect all wires. First, display the number of cuts needed. Then display the cuts, one per line in the format x_1 y_1 x_2 y_2 for the cut between (x_1, y_1) and

(x_2, y_2). Each cut has to start and end on different sides of the door. Cuts cannot start or end closer than 10^{-6} to any wire anchor location or any corner of the door.

Cuts may be displayed in any order. The start and end locations of each cut may be displayed in either order. If there are multiple sets of cuts with the same minimum size, display any of them.

Examples

standard input	standard output

`4 4 6`
`0 1 4 4`
`0 5 2 0`
`0 3 3 6`
`2 6 4 2`
	`1`
	`0 4 4 3`

`5 4 6`
`0 2 2 0`
`0 3 2 6`
`1 6 3 0`
`1 0 4 4`
`3 6 4 2`
	`2`
	`0 4 4 4.5`
	`0 1 4 1`

Problem I. Triangles

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	1024 megabytes

For your trip to Beijing, you have brought plenty of puzzle books, many of them containing challenges like the following: how many triangles can be found in Figure I.1?

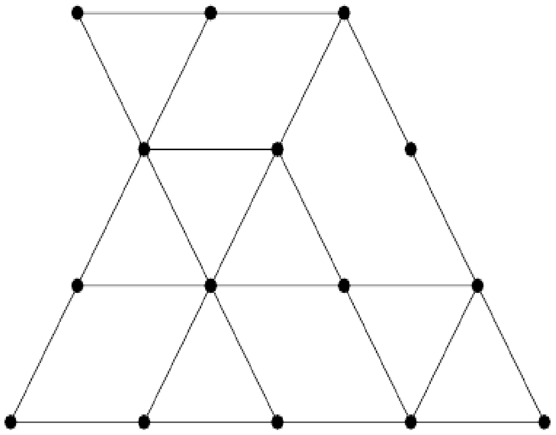


Figure I.1: Illustration of Sample Input 2.

While these puzzles keep your interest for a while, you quickly get bored with them and instead start thinking about how you might solve them algorithmically. Who knows, maybe a problem like that will actually be used in this year's contest. Well, guess what? Today is your lucky day!

Input

The first line of input contains two integers r and c ($1 \le r \le 3000$, $1 \le c \le 6000$), specifying the picture size, where r is the number of rows of vertices and c is the number of columns. Following this are $2r - 1$ lines, each of them having at most $2c - 1$ characters. Odd lines contain grid vertices (represented as lowercase `x` characters) and zero or more horizontal edges, while even lines contain zero or more diagonal edges. Specifically, picture lines with numbers $4k+1$ have vertices in positions $1, 5, 9, 13, \ldots$ while lines with numbers $4k + 3$ have vertices in positions $3, 7, 11, 15, \ldots$. All possible vertices are represented in the input (for example, see how Figure I.1 is represented in Sample Input 2). Horizontal edges connecting neighboring vertices are represented by three dashes. Diagonal edges are represented by a single forward slash ('/') or backslash ('\') character. The edge characters will be placed exactly between the corresponding vertices. All other characters will be space characters. Note that if any input line could contain trailing whitespace, that whitespace may be omitted.

Output

Display the number of triangles (of any size) formed by grid edges in the input picture.

Examples

standard input	standard output

`3 3`
`x---x`
` \ /`
`  x`
` / \`
`x   x`
	`1`

`4 10`
`x   x---x---x   x`
`     \ /   / \`
`  x   x---x   x   x`
`     / \ / \   \`
`x   x---x---x---x`
`   /   / \   \ / \`
`  x---x---x---x---x`
	`12`

Problem J. Uncrossed Knight's Tour

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

A well-known puzzle is to "tour" all the squares of an 8×8 chessboard using a knight, which is a piece that can move only by jumping one square in one direction and two squares in an orthogonal direction. The knight must visit every square of the chessboard, without repeats, and then return to its starting square. There are many ways to do this, and the chessboard size is manageable, so it is a reasonable puzzle for a human to solve.

However, you have access to a computer, and some coding skills! So, we will give you a harder version of this problem on a rectangular $m \times n$ chessboard with an additional constraint: the knight may never cross its own path. If you imagine its path consisting of straight line segments connecting the centers of squares it jumps between, these segments must form a simple polygon; that is, no two segments intersect or touch, except that consecutive segments touch at their common end point. This constraint makes it impossible to visit every square, so instead you must maximize the number of squares the knight visits. We keep the constraint that the knight must return to its starting square. Figure J.1 shows an optimal solution for the first sample input, a 6×6 board.

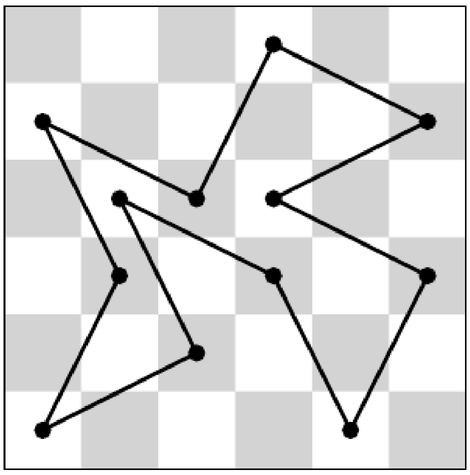


Figure J.1: An optimal solution for a 6 ×6 board.

Input

The input consists of a single line containing two integers m ($1 \le m \le 8$) and n ($1 \le n \le 10^{15}$), giving the dimensions of the rectangular chessboard.

Output

Display the largest number of squares that a knight can visit in a tour on an $m \times n$ chessboard that does not cross its path. If no such tour exists, display 0.

Examples

standard input	standard output
6 6	12
8 3	6
7 20	80
2 6	0

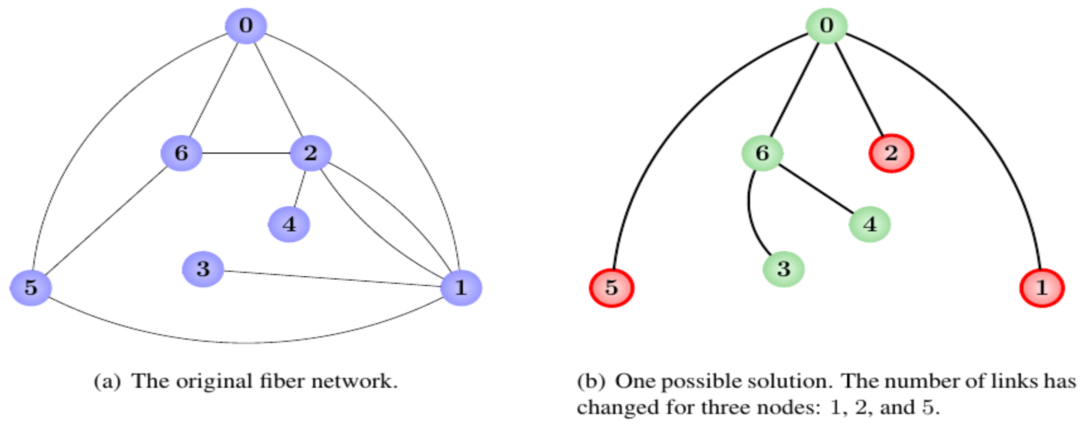
Problem K. Wireless is the New Fiber

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

A new type of unbounded-bandwidth wireless communication has just been tested and proved to be a suitable replacement for the existing, fiber-based communications network, which is struggling to keep up with traffic growth. You have been charged with deciding the layout of the new communications network. The current communications network consists of a set of nodes (which route messages), and links of fiber, each of which connects two different nodes. For each pair of nodes, there exists at least one way (but possibly more, for bandwidth purposes) to travel along the fiber between the two.

The new communications network will not have any fiber. Instead, it will have wireless links, each connecting two nodes. These links have unbounded bandwidth but are expensive, so it has been decided that as few of these links will be built as possible to provide connectivity; for each pair of nodes there should be exactly one way to travel between them along the wireless links. Moreover, you discovered that the nodes have each been built with a particular number of connections in mind. For each node, if it will be connected to a different number of links than it is today, it will have to be reorganized, and that is costly.

Your task is to design the new network so that it has precisely one path between each pair of nodes while minimizing the number of nodes that do not have the same number of connections as in the original network.



(a) The original fiber network.

(b) One possible solution. The number of links has changed for three nodes: 1, 2, and 5.

Figure K.1: Illustration of Sample Input 1.

Input

The input begins with a line containing two integers n ($2 \le n \le 10^4$) and m ($1 \le m \le 10^5$), denoting the number of nodes and the number of fiber links in the existing network. The nodes are numbered from 0 to $n-1$. Each of the next m lines contains two distinct integers a_i and b_i , denoting the fact that the i^{th} fiber link connects nodes numbered a_i and b_i . It is guaranteed that for each pair of nodes there exists at least one path connecting the two nodes. Any pair of nodes may have more than one fiber link connecting them.

Output

Display the smallest number of nodes for which the number of connected links needs to change. Starting on the next line, display a system of connections in the same format as the input. That is, display a line containing the number of nodes (this will be the same as in the input) and the number of wireless links, and then on subsequent lines descriptions of the links. If more than one layout is possible, any valid layout will be accepted.

Examples

standard input	standard output
7 11	3
0 1	7 6
0 2	0 1
0 5	0 2
0 6	0 5
1 3	0 6
2 4	3 6
1 2	4 6
1 2	
1 5	
2 6	
5 6	
4 3	0
0 1	4 3
2 1	2 1
2 3	1 3
	0 2