Problem A
Airport Construction
Time limit: 2 seconds

The tropical island nation of Piconesia is famous for its beautiful beaches, lush vegetation, cocoa and coffee plantations, and wonderful weather all year round. This paradise is being considered as a future location for the World Finals of the ACM International Collegiate Programming Contest (or at the very least a vacation spot for the executive council). There is only one small problem: the island is really hard to reach.

Currently, the fastest way to reach the island takes three days from the nearest airport, and uses a combination of fishing boat, oil tanker, kayak, and submarine. To make attending the ICPC World Finals slightly easier and to jump-start the island's tourism business, Piconesia is planning to build its first airport.

Since longer landing strips can accommodate larger airplanes, Piconesia has decided to build the longest possible landing strip on their island. Unfortunately, they have been unable to determine where this landing strip should be located. Maybe you can help?

For this problem we model the boundary of Piconesia as a polygon. Given this polygon, you need to compute the length of the longest landing strip (i.e., straight line segment) that can be built on the island. The landing strip must not intersect the sea, but it may touch or run along the boundary of the island. Figure A.1 shows an example corresponding to the first sample input.

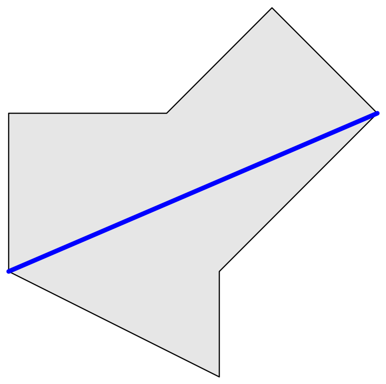


Figure A.1: The island modeled as a polygon. The longest possible landing strip is shown as a thick line.

Input

The input starts with a line containing an integer n ($3 \leq n \leq 200$) specifying the number of vertices of the polygon. This is followed by n lines, each containing two integers x and y ($|x|, |y| \leq 10^6$) that give the coordinates (x, y) of the vertices of the polygon in counter-clockwise order. The polygon is simple, i.e., its vertices are distinct and no two edges of the polygon intersect or touch, except that consecutive edges touch at their common vertex. In addition, no two consecutive edges are collinear.

Output

Display the length of the longest straight line segment that fits inside the polygon, with an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1

7
0 20
40 0
40 20
70 50
50 70
30 50
0 50
	76.157731059

Sample Input 2	Sample Output 2

3
0 2017
-2017 -2017
2017 0
	4510.149110617

Problem B
Get a Clue!
Time limit: 4 seconds

Developed in the 1940s in the United Kingdom, the game of Cluedo is one of the most popular board games in the world. The object of the game is to determine who murdered Mr. Body, which weapon was used to murder him, and where the murder took place. The game uses a set of cards representing six persons (labeled A, B, . . . , F), six weapons (labeled G, H, . . . , L) and nine rooms (labeled M, N, . . . , U). At the start of the game, one person card, one weapon card, and one room card are selected at random and removed from the deck so no one can see them – they represent the murderer, the murder weapon, and the murder location. The remaining 18 cards are shuffled and dealt to the players, starting with player 1, then to her right player 2, and so on. Some players may end up with one more card than others. For the purposes of this problem there are four players, so the person to the right of player 4 is player 1.

The rest of the game is spent searching for clues. Players take turns, starting with player 1 and moving to the right. A turn consists of making a *suggestion* (consisting of a murder suspect, a weapon, and a room) and asking other players if they have any evidence that refutes the suggestion. For example, you might say to another player "I believe the murderer was person A, using weapon L, in room T." If the other player is holding exactly one of these cards, that player must show you (and only you) that card. If they have more than one such card, they can show you any one of them.

When making a suggestion, you must first ask the person to your right for any evidence. If they have none, you continue with the person on their right, and so on, until someone has evidence, or no one has any of the cards in your suggestion.

Many times you can gain information even if you are not the person making the suggestion. Suppose, in the above example, you are the third player and have cards A and T. If someone else shows evidence to the suggester, you know that it must be weapon card L. Keeping track of suggestions and who gave evidence at each turn is an important strategy when playing the game.

To win the game, you must make an *accusation*, where you state your final guess of the murderer, weapon, and room. After stating your accusation, you check the three cards that were set aside at the start of the game – if they match your accusation, you win! Needless to say, you want to be absolutely sure of your accusation before you make it.

Here is your problem. You are player 1. Given a set of cards dealt to you and a history of suggestions and evidence, you need to decide how close you are to being able to make an accusation.

Input

The input starts with an integer n ($1 \leq n \leq 50$), the number of suggestions made during the game. Following this is a line containing the five cards you are dealt, all uppercase letters in the range '`A`'. . . '`U`'. The remaining n lines contain one suggestion per line. Each of these lines starts with three characters representing the suggestion (in the order person, weapon, room), followed by the responses of up to three players, beginning with the player to the right of the player making the suggestion. If a player presents no evidence, a '`-`' (dash) is listed; otherwise an "evidence character" is listed. If the specific evidence card is seen by you (either because you provided it or you were the person receiving the evidence) then the evidence character identifies that card; otherwise the evidence character is ‘⋆’. Note that only the last response can be an evidence character. All characters are separated by single spaces. Only valid suggestion/response sequences appear in the input.

Output

Display a three character string identifying the murderer, the murder weapon, and the room. If the murderer can be identified, use the appropriate letter for that person; otherwise use ‘?’. Do the same for the murder weapon and the room.

Sample Input 1	Sample Output 1

```
1
B I P C F
A G M - - -
```
	`AGM`

Sample Input 2	Sample Output 2

```
2
A B C D H
F G M M
F H M - ⋆
```
	`E??`

Sample Input 3	Sample Output 3

```
3
A C M S D
B G S - G
A H S - - S
C J S ⋆
```
	`???`

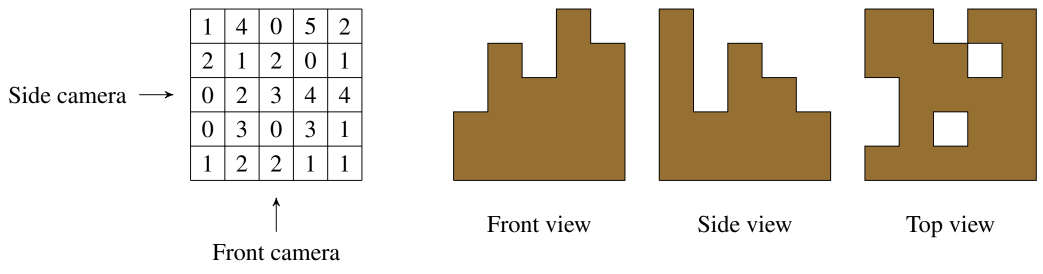
Problem C
Mission Improbable
Time limit: 1 second

It is a sunny day in spring and you are about to meet Patrick, a close friend and former partner in crime. Patrick lost most of his money betting on programming contests, so he needs to pull off another job. For this he needs your help, even though you have retired from a life of crime. You are reluctant at first, as you have no desire to return to your old criminal ways, but you figure there is no harm in listening to his plan.

There is a shipment of expensive consumer widgets in a nearby warehouse and Patrick intends to steal as much of it as he can. This entails finding a way into the building, incapacitating security guards, passing through various arrays of laser beams – you know, the usual heist techniques. However, the heart of the warehouse has been equipped with a security system that Patrick cannot disable. This is where he needs your help.

The shipment is stored in large cubical crates, all of which have the same dimensions. The crates are stacked in neat piles, forming a three-dimensional grid. The security system takes pictures of the piles once per hour using three cameras: a front camera, a side camera and a top camera. The image from the front camera shows the height of the tallest pile in each column, the image from the side camera shows the height of the tallest pile in each row, and the image from the top camera shows whether or not each pile is empty. If the security system detects a change in any of the images, it sounds an alarm.

Once Patrick is inside, he will determine the heights of the piles and send them to you. Figure C.1 shows a possible layout of the grid and the view from each of the cameras.



1	4	0	5	2
2	1	2	0	1
0	2	3	4	4
0	3	0	3	1
1	2	2	1	1

Figure C.1: Grid of heights and the corresponding camera views.

1	4	0	5	1
2	1	1	0	1
0	1	3	1	4
0	3	0	1	1
2	1	1	1	1

Figure C.2: Possible grid of heights after the heist

Patrick wants to steal as many crates as possible. Since he cannot disable the security system, he plans to fool it by arranging the remaining crates into piles so that the next set of camera images are the same. In the above example, it is possible to steal nine crates. Figure C.2 shows one possible post-heist configuration that appears identical to the security system.

Patrick asks you to help him determine the maximum number of crates that can be stolen while leaving a configuration of crates that will fool the security system. Will you help him pull off this final job?

Input

The first line of input contains two integers r ($1 \leq r \leq 100$) and c ($1 \leq c \leq 100$), the number of rows and columns in the grid, respectively. Each of the following r lines contains c integers, the heights (in crates) of the piles in the corresponding row. All heights are between 0 and 10^9 inclusive.

Output

Display the maximum number of crates that can be stolen without being detected.

Sample Input 1

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

Sample Output 1

```
9
```

Sample Input 2

```
2 3
50 20 3
20 10 3
```

Sample Output 2

```
30
```

Problem D
Money for Nothing
Time limit: 5 seconds

In this problem you will be solving one of the most profound challenges of humans across the world since the beginning of time – how to make lots of money.

You are a middleman in the widget market. Your job is to buy widgets from widget producer companies and sell them to widget consumer companies. Each widget consumer company has an open request for one widget per day, until some end date, and a price at which it is willing to buy the widgets. On the other hand, each widget producer company has a start date at which it can start delivering widgets and a price at which it will deliver each widget.

Due to fair competition laws, you can sign a contract with only one producer company and only one consumer company. You will buy widgets from the producer company, one per day, starting on the day it can start delivering, and ending on the date specified by the consumer company. On each of those days you earn the difference between the producer's selling price and the consumer's buying price.

Your goal is to choose the consumer company and the producer company that will maximize your profits.

Input

The first line of input contains two integers m and n ($1 \le m, n \le 500\,000$) denoting the number of producer and consumer companies in the market, respectively. It is followed by m lines, the i^{th} of which contains two integers p_i and d_i ($1 \le p_i, d_i \le 10^9$), the price (in dollars) at which the i^{th} producer sells one widget and the day on which the first widget will be available from this company. Then follow n lines, the j^{th} of which contains two integers q_j and e_j ($1 \le q_j, e_j \le 10^9$), the price (in dollars) at which the j^{th} consumer is willing to buy widgets and the day immediately after the day on which the last widget has to be delivered to this company.

Output

Display the maximum total number of dollars you can earn. If there is no way to sign contracts that gives you any profit, display `0`.

Sample Input 1

```
2 2
1 3
2 1
3 5
7 2
```

Sample Output 1

```
5
```

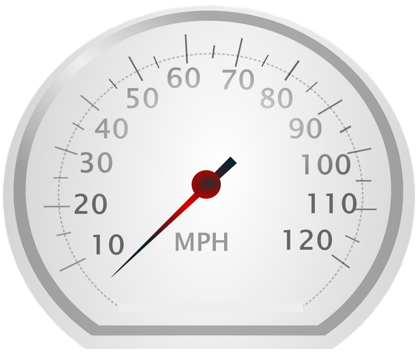
Sample Input 2

```
1 2
10 10
9 11
11 9
```

Sample Output 2

```
0
```

Problem E
Need for Speed
Time limit: 1 second



Sheila is a student and she drives a typical student car: it is old, slow, rusty, and falling apart. Recently, the needle on the speedometer fell off. She glued it back on, but she might have placed it at the wrong angle. Thus, when the speedometer reads s, her true speed is $s + c$, where c is an unknown constant (possibly negative).

Sheila made a careful record of a recent journey and wants to use this to compute c. The journey consisted of n segments. In the i^{th} segment she traveled a distance of d_i and the speedometer read s_i for the entire segment. This whole journey took time t. Help Sheila by computing c.

Note that while Sheila's speedometer might have negative readings, her true speed was greater than zero for each segment of the journey.

Input

The first line of input contains two integers n ($1 \leq n \leq 1\,000$), the number of sections in Sheila's journey, and t ($1 \leq t \leq 10^6$), the total time. This is followed by n lines, each describing one segment of Sheila's journey. The i^{th} of these lines contains two integers d_i ($1 \leq d_i \leq 1\,000$) and s_i ($|s_i| \leq 1\,000$), the distance and speedometer reading for the i^{th} segment of the journey. Time is specified in hours, distance in miles, and speed in miles per hour.

Output

Display the constant c in miles per hour. Your answer should have an absolute or relative error of less than 10^{-6}.

Sample Input 1	Sample Output 1

3 5
4 -1
4 0
10 3
	3.000000000

Sample Input 2	Sample Output 2

4 10
5 3
2 2
3 6
3 1
	-0.508653377

Problem F
Posterize
Time limit: 2 seconds

Pixels in a digital picture can be represented with three integers in the range 0 to 255 that indicate the intensity of the red, green, and blue colors. To compress an image or to create an artistic effect, many photo-editing tools include a "posterize" operation which works as follows. Each color channel is examined separately; this problem focuses only on the red channel. Rather than allow all integers from 0 to 255 for the red channel, a posterized image allows at most k integers from this range. Each pixel's original red intensity is replaced with the nearest of the allowed integers. The photo-editing tool selects a set of k integers that minimizes the *sum of the squared errors* introduced across all pixels in the original image. If there are n pixels that have original red values $r_1, \ldots, r_n$, and k allowed integers $v_1, \ldots, v_k$, the sum of squared errors is defined as

$$\sum_{i=1}^{n} \min_{1 \le j \le k} (r_i - v_j)^2 .$$

Your task is to compute the minimum achievable sum of squared errors, given parameter k and a description of the red intensities of an image's pixels.

Input

The first line of the input contains two integers d ($1 \le d \le 256$), the number of distinct red values that occur in the original image, and k ($1 \le k \le d$), the number of distinct red values allowed in the posterized image. The remaining d lines indicate the number of pixels of the image having various red values. Each such line contains two integers r ($0 \le r \le 255$) and p ($1 \le p \le 2^{26}$), where r is a red intensity value and p is the number of pixels having red intensity r. Those d lines are given in increasing order of red value.

Output

Display the sum of the squared errors for an optimally chosen set of k allowed integer values.

Sample Input 1

```
2 1
50 20000
150 10000
```

Sample Output 1

```
66670000
```

Sample Input 2

```
2 2
50 20000
150 10000
```

Sample Output 2

```
0
```

Sample Input 3

```
4 2
0 30000
25 30000
50 30000
255 30000
```

Sample Output 3

```
37500000
```

Problem G
Replicate Replicate Rfplicbte
Time limit: 3 seconds

The owner of the Automatic Cellular Manufacturing corporation has just patented a new process for the mass production of identical parts. Her approach uses a two-dimensional lattice of two-state cells, each of which is either "empty" or "filled." The exact details are, of course, proprietary.

Initially, a set of cells in the lattice is filled with a copy of the part that is to be reproduced. In a sequence of discrete steps, each cell in the lattice simultaneously updates its state by examining its own state and those of its eight surrounding neighbors. If an odd number of these nine cells are filled, the cell's state in the next time step will be filled, otherwise it will be empty. Figure G.1 shows several steps in the replication process for a simple pattern consisting of three filled cells.

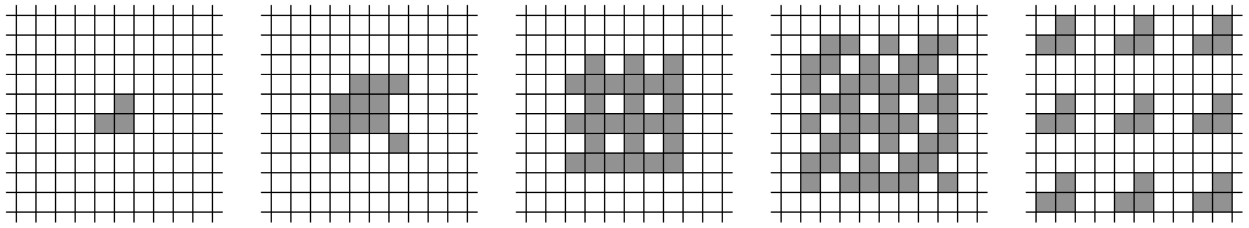


Figure G.1: The replication process.

However, a bug has crept into the process. After each update step, one cell in the lattice might spontaneously flip its state. For instance, Figure G.2 shows what might happen if a cell flipped its state after the first time step and another flipped its state after the third time step.

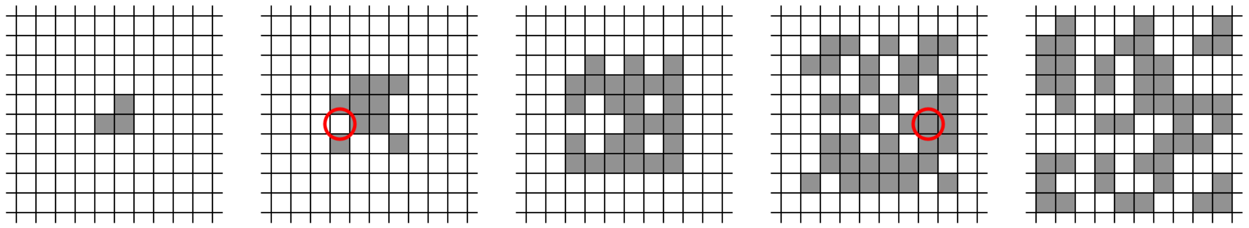


Figure G.2: Errors in the replication process. This figure corresponds to Sample Input 1.

Unfortunately, the original patterns were lost, and only the (possibly corrupted) results of the replication remain. Can you write a program to determine a smallest possible nonempty initial pattern that could have resulted in a given final pattern?

Input

The first line of input contains two integers w ($1 \leq w \leq 300$) and h ($1 \leq h \leq 300$), where w and h are the width and height of the bounding box of the final pattern. Following that are h lines, each containing w characters, giving the final pattern. Each character is either '`.`' (representing an empty cell) or '`#`' (representing a filled cell). There is at least one filled cell in the first row, in the last row, in the first column, and in the last column.

Output

Display a minimum-size nonempty pattern that could have resulted in the given pattern, assuming that at each stage of the replication process at most one cell spontaneously changed state. The size of a pattern is the area of its bounding box. If there is more than one possible minimum-size nonempty starting pattern, any one will be accepted. Use the character '`.`' for empty cells and '`#`' for filled cells. Use the minimum number of rows and columns needed to display the pattern.

Sample Input 1

```
10 10
.#...#...#
##..##..##
##.#.##...
##.#.##...
.#...#####
...##..#.#
......###.
##.#.##...
#..#..#..#
##..##..##
```

Sample Output 1

```
.#
##
```

Sample Input 2

```
8 8
##..#.##
#.####.#
.#.#.#..
.##.#.##
.#.#.#..
.##.#.##
#..#.###
##.#.##.
```

Sample Output 2

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

Sample Input 3

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

Sample Output 3

```
#
```

Problem H
Scenery

Time limit: 6 seconds

Images by John Fowler, Carol Highsmith, and Richard Woodland

You have decided to spend a day of your trip to Rapid City taking photographs of the South Dakota Badlands, which are renowned for their spectacular and unusual land formations. You are an amateur photographer, yet very particular about lighting conditions.

After some careful research, you have located a beautiful location in the Badlands, surrounded by picturesque landscapes. You have determined a variety of features that you wish to photograph from this location. For each feature you have identified the earliest and latest time of day at which the position of the sun is ideal. However, it will take quite a bit of time to take each photograph, given the need to reposition the tripod and camera and your general perfectionism. So you are wondering if it will be possible to successfully take photographs of all these features in one day.

Input

The first line of the input contains two integers n ($1 \le n \le 10^4$) and t ($1 \le t \le 10^5$), where n is the number of desired photographs and t is the time you spend to take each photograph. Following that are n additional lines, each describing the available time period for one of the photographs. Each such line contains two nonnegative integers a and b, where a is the earliest time that you may begin working on that photograph, and b is the time by which the photograph must be completed, with $a + t \le b \le 10^9$.

Output

Display `yes` if it is possible to take all n photographs, and `no` otherwise.

Sample Input 1

```
2 10
0 15
5 20
```

Sample Output 1

```
yes
```

Sample Input 2

```
2 10
1 15
0 20
```

Sample Output 2

```
no
```

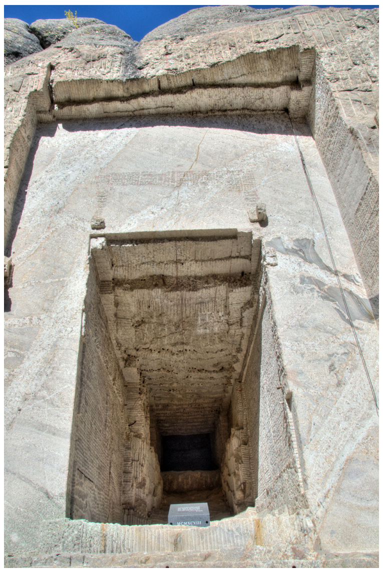
Sample Input 3

```
2 10
5 30
10 20
```

Sample Output 3

```
yes
```

Problem I
Secret Chamber at Mount Rushmore
Time limit: 1 second



By now you have probably heard that there is a spectacular stone sculpture featuring four famous U.S. presidents at Mount Rushmore. However, very few people know that this monument contains a secret chamber. This sounds like something out of a plot of a Hollywood movie, but the chamber really exists. It can be found behind the head of Abraham Lincoln and was designed to serve as a Hall of Records to store important historical U.S. documents and artifacts. Historians claim that the construction of the hall was halted in 1939 and the uncompleted chamber was left untouched until the late 1990s, but this is not the whole truth.

In 1982, the famous archaeologist S. Dakota Jones secretly visited the monument and found that the chamber actually was completed, but it was kept confidential. This seemed suspicious and after some poking around, she found a hidden vault and some documents inside. Unfortunately, these documents did not make any sense and were all gibberish. She suspected that they had been written in a code, but she could not decipher them despite all her efforts.

Earlier this week when she was in the area to follow the ACM-ICPC World Finals, Dr. Jones finally discovered the key to deciphering the documents, in Connolly Hall of SDSM&T. She found a document that contains a list of translations of letters. Some letters may have more than one translation, and others may have no translation. By repeatedly applying some of these translations to individual letters in the gibberish documents, she might be able to decipher them to yield historical U.S. documents such as the Declaration of Independence and the Constitution. She needs your help.

You are given the possible translations of letters and a list of pairs of original and deciphered words. Your task is to verify whether the words in each pair match. Two words match if they have the same length and if each letter of the first word can be turned into the corresponding letter of the second word by using the available translations zero or more times.

Input

The first line of input contains two integers m $(1 \leq m \leq 500)$ and n $(1 \leq n \leq 50)$, where m is the number of translations of letters and n is the number of word pairs. Each of the next m lines contains two distinct space-separated letters a and b, indicating that the letter a can be translated to the letter b. Each ordered pair of letters (a, b) appears at most once. Following this are n lines, each containing a word pair to check. Translations and words use only lowercase letters '`a`'–'`z`', and each word contains at least 1 and at most 50 letters.

Output

For each pair of words, display `yes` if the two words match, and `no` otherwise.

Sample Input 1

```
9 5
c t
i r
k p
o c
r o
t e
t f
u h
w p
we we
can the
work people
it of
out the
```

Sample Output 1

```
yes
no
no
yes
yes
```

Sample Input 2

```
3 3
a c
b a
a b
aaa abc
abc aaa
acm bcm
```

Sample Output 2

```
yes
no
yes
```

Problem J
Son of Pipe Stream
Time limit: 5 seconds

Two years ago, you helped install the nation's very first Flubber pipe network in your hometown, to great success. Polls show that everyone loves having their own Flubber dispenser in their kitchen, and now a few enterprising citizens have discovered a use for it. Apparently Flubber, when mixed with water, can help extinguish fires! This is a very timely discovery, as out-of-control fires have lately been surprisingly common.

Your hometown's city council would like to make use of this property of Flubber by creating the Flubber/water mixture at a centrally located station. This station, which is called the Flubber Department (FD) will also have specialized employees trained to travel to the locations of fires and make use of their processed Flubber to control the blazes.

The pipes are already in place all around the city. You are given a layout of the pipes, and must determine how to route Flubber from the Flubber factory and water from a local source through the pipes to the FD.

Note that both Flubber and water will be flowing through the same network of pipes, perhaps even the same pipe. All pipes are bidirectional, but Flubber and water cannot move in opposite directions through the same pipe. Furthermore, if both liquids are sent in the same direction through the same pipe, they will inevitably mix. Therefore the nodes in the network have been equipped with special membranes and filters that enable you to separate and reorganize all incoming mixtures as you see fit. The network is a closed system, so the total rate of each fluid going into a node must equal the total rate of that fluid going out, except at the source of that fluid and the destination (the FD).

Each pipe has a certain capacity. Flubber, being somewhat sluggish, has a viscosity value v, so a pipe that can transport v liters/second of water can transport only 1 liter/second of Flubber. The pipe's capacity scales linearly for mixtures of the two. To be precise, if c denotes the water capacity of the pipe and f and w are the rates of Flubber and water moving through the pipe (all measured in liters/second), then the capacity constraint is given by the inequality $v \cdot f + w \le c$.

Your main concern is balancing the mixture that reaches the FD. You would like as much total liquid as possible, but you also need a sufficient amount of water – because undiluted Flubber is highly flammable – and a sufficient amount of Flubber – because it would not be much of a "Flubber Department" without Flubber! You have come up with a formula to measure the "value" of the final mixture: $F^a \cdot W^{1-a}$, where F is the rate of incoming Flubber in liters/second, W is the rate of incoming water in liters/second, and a is a given constant between 0 and 1.

Determine the maximum value of $F^a \cdot W^{1-a}$ that can be achieved and how to route the Flubber and water to achieve it.

Input

The input starts with a line containing the number of locations n ($3 \le n \le 200$), the number of pipes p ($n-1 \le p \le \frac{1}{2}n(n-1)$), and the real values v ($1 \le v \le 10$) and a ($0.01 \le a \le 0.99$). Locations are numbered from 1 to n; 1 is the Flubber factory, 2 is the water source, and 3 is the FD. The real values have at most 10 digits after the decimal point.

The following p lines each describe one pipe. Each line contains two integers j and k ($1 \leq j < k \leq n$), giving the locations connected by the pipe, and an integer c ($1 \leq c \leq 10$), giving the water capacity of the pipe in liters/second.

No two pipes connect the same pair of locations. Furthermore, it is guaranteed that the network is connected.

Output

First, for each pipe (in the order given in the input), display two values: the rate of Flubber moving through it, and the rate of water moving through it (negative if the liquid is moving from k to j), such that $F^a \cdot W^{1-a}$ is maximized. Then display that maximum value accurate to within an absolute error of 10^{-4}.

If there are multiple solutions, any one will be accepted. All constraints (not sending Flubber and water in opposite directions along the same pipe, flow conservation, pipe capacities, and consistency between the constructed solution and its claimed value) must be satisfied within an absolute error of 10^{-4}.

Sample Input 1

```
6 6 3.0 0.66
2 4 8
4 6 1
3 6 1
4 5 5
1 5 7
3 5 3
```

Sample Output 1

```
0.000000000 1.360000000
0.000000000 1.000000000
0.000000000 -1.000000000
0.000000000 0.360000000
0.880000000 0.000000000
-0.880000000 -0.360000000
1.02037965897
```

Sample Input 2

```
5 5 1.0 0.5
1 2 10
2 3 10
3 4 10
4 5 10
3 5 10
```

Sample Output 2

```
5 0
5 5
4.2 3.14159
4.2 3.14159
-4.2 -3.14159
5
```

Problem K
Tarot Sham Boast
Time limit: 2 seconds

Curse your rival! Every year at the annual Rock Paper Scissors tournament, you have made it to the final match. (Your Rock technique is unmatched, and your Paper cuts to the bone! Your Scissors need a little work, though.) But every year, he defeats you, even though his moves appear entirely random! And he claims to the press that he simply cannot be beaten. What is his secret?

Fortunately, you think you have figured it out. This year, just before the tournament, you caught him visiting various shamans around town. Aha! He is using the supernatural against you! You figured two can play at this game. So you went and visited a set of fortune-tellers, who have each used a Tarot deck to predict a sequence that your rival will end up using, sometime during the match.

However, your initial excitement has passed, and now you are feeling a little silly. This cannot possibly work, right? In the end it feels like you have paid good money for a fraudulent, random set of predictions. Oh well; you might as well keep an eye out for some of them during the match. But which predictions will you use?

In the final match, you and your rival will play n rounds of Rock Paper Scissors. In each round, your rival and you will both choose one of the three options (Rock, Paper, or Scissors). Based on your selections, a winner of the round will be determined (exactly *how* is irrelevant to this problem).

Given the length of the final match and the various predictions, sort them in order of how likely they are to appear sometime during the match as a contiguous sequence of options chosen by your rival, assuming he is choosing his symbol in each round independently and uniformly at random.

Input

The first line of input contains two integers n ($1 \leq n \leq 10^6$), the number of rounds in the final match, and s ($1 \leq s \leq 10$), the number of sequences. The remaining s lines each describe a prediction, consisting of a string of characters ‘R’, ‘P’, and ‘S’. All predictions have the same length, which is between 1 and n characters long, inclusive, and no longer than 10^5.

Output

Display all of the predictions, sorted by decreasing likelihood of appearance sometime during the final match. In the case of tied predictions, display them in the same order as in the input.

Sample Input 1	Sample Output 1
3 4 PP RR PS SS	PS PP RR SS

Sample Input 2	Sample Output 2

`20 3`
`PRSPS`
`SSSSS`
`PPSPP`
	`PRSPS`
	`PPSPP`
	`SSSSS`

Problem L
Visual Python++
Time limit: 5 seconds

In the recently proposed Visual Python++ programming language, a block of statements is represented as a rectangle of characters with top-left corner in row r_1 and column c_1, and bottom-right corner in row r_2 and column c_2. All characters at locations (r, c) with $r_1 \leq r \leq r_2$ and $c_1 \leq c \leq c_2$ are then considered to belong to that block. Among these locations, the ones with $r = r_1$ or $r = r_2$ or $c = c_1$ or $c = c_2$ are called a border.

Statement blocks can be nested (rectangles contained in other rectangles) to an arbitrary level. In a syntactically correct program, every two statement blocks are either nested (one contained in the other) or disjoint (not overlapping). In both cases, their borders may not overlap.

Programmers are not expected to draw the many rectangles contained in a typical program – this takes too long, and Visual Python++ would not have a chance to become the next ICPC programming language. So a programmer only has to put one character '┌' in the top-left corner of the rectangle and one character '┘' in the bottom-right corner. The parser will automatically match up the appropriate corners to obtain the nesting structure of the program.

Your team has just been awarded a five-hour contract to develop this part of the parser.

Input

The first line of the input contains an integer n ($1 \leq n \leq 10^5$), the number of corner pairs. Each of the next n lines contains two integers r and c ($1 \leq r, c \leq 10^9$), specifying that there is a top-left corner in row r and column c of the program you are parsing. Following that are n lines specifying the bottom-right corners in the same way. All corner locations are distinct.

Output

Display n lines, each containing one integer. A number j in line i means that the i^{th} top-left corner and the j^{th} bottom-right corner form one rectangle. Top-left and bottom-right corners are each numbered from 1 to n in the order they appear in the input. The output must be a permutation of the numbers from 1 to n such that the matching results in properly nested rectangles. If there is more than one valid matching, any one will be accepted. If no such matching exists, display `syntax error`.

Sample Input 1

```
2
4 7
9 8
14 17
19 18
```

Sample Output 1

```
2
1
```

Sample Input 2

```
2
4 7
14 17
9 8
19 18
```

Sample Output 2

```
1
2
```

Sample Input 3

```
2
4 8
9 7
14 18
19 17
```

Sample Output 3

```
syntax error
```

Sample Input 4

```
3
1 1
4 8
8 4
10 6
6 10
10 10
```

Sample Output 4

```
syntax error
```