Problem A. Arrays

Input file:	`arrays.in`
Output file:	`arrays.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

All the companies interview their candidates. *Facepalm* also does it. The most favourite questions during the interview are various array problems. One example of such problem is to find the minimal element in an array. Of course, there are many other problems.

This problem is also an array problem. When you solve it, you will perhaps find yourself one step closer to working for *Facepalm*.

The problem is the following. You are given n arrays of integers and an integer k. You should select no more than k elements from each of these arrays so that all the selected elements from all arrays are pairwise distinct and their sum is as large as possible.

Input

The first line of input contains two integers n and k ($1 \leq n, k \leq 50$). Each of the next n lines contains one array description. The first number in an array description is an integer s, the number of elements in the array ($1 \leq s \leq 50$). It is followed by s integers which constitute the array itself (integers are in the range from 1 to 1000).

Output

On the first line, print one integer: the sum of selected elements. Each of the next n lines must describe the elements taken from one array. i-th of these lines must start with an integer m_i, the number of elements selected from i-th array. It must be followed by m_i integers which are the selected elements from that array. Elements can be printed in any order. If there is more than one solution, you may output any of them.

Examples

arrays.in	arrays.out

2 3
6 1 2 3 4 5 6
6 1 2 3 4 5 6
	21
	3 4 5 6
	3 1 2 3

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

Problem B. Goondex

Input file:	`goondex.in`
Output file:	`goondex.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

The *Facepalm* office is located in the area near all the leading software companies' offices. One of these companies is *Goondex*. The employees of this company always say: "What don't we do, bro?".

This company is famous for its search engine. Everybody is familiar with it. This engine has lots of interesting features. One of these features is the guessing system for user queries.

Now *Facepalm* is trying to create a similar search engine for their own server. They want to find out how this guessing system works. They worked hard and wrote down several rules they thought were important. Here they are.

Each query contains exactly one word consisting of lowercase English letters. *Facepalm* has got access to a database consisting of different words that users try to find. Each of these words has a rating: the more popular the word, the higher the rating. Each new query increases the rating of the word in this query by one.

The guessing system works as follows. The user is forming an input string to the search engine. Initially, the input string is empty. Normally, the user would just enter the word he has in mind character by character from left to right, and these characters are appended to the input string. Sometimes after pressing a key, the user sees a hint from the guessing system based on the input string. The hint is chosen from the set of words in the database whose prefix is equal to the input string and length is strictly greater than the length of the input string. If there are several such words, the one with the highest rating is given. If there are still several choices, the guessing system chooses the one that comes earlier lexicographically. In case there are no words which meet the above criteria, or the user did not yet enter anything, the guessing system will give no hint.

Now let's take a look at the user's actions. If the user sees a hint that is exactly the word he is going to enter, he presses the "Enter" key and the input string is finalized. Otherwise, if he sees a hint that is a prefix of his query word, he presses the "Space" key, the input string becomes equal to the hint, and the guessing system is asked for a hint again. Whenever there is no hint, the user just enters the next character of the word. The process continues until the "Enter" key is pressed. Additionally, there is a "magical Enter" key which discards the hint and finalizes the input string without it. Obviously, the "Enter" key will be pressed only once and only in the end of the query.

After the user presses the "Enter" key, the rating of the word in the query is increased by one. If there was no such word in the database, it is added, and its rating is set to one.

Now *Facepalm* is interested in checking if this guessing system is good indeed. To answer this question, you are given a number of queries, and you should calculate two numbers. The first number is the total number of keystrokes, i. e., the number of times the user presses a chacacter key, the "Enter" key, or the "Space" key. The second one is the total number of characters in all the queries plus the number of queries. That would be the number of keystrokes if there was no guessing system at all. Obviously, if the first number is not less than the second one, such guessing system is not worth implementing.

Input

The first line of input contains the number of words n in the database before all queries ($1 \le n \le 10\,000$). The next n lines contain the words along with their ratings. Ratings are integers from 1 to 1000. All the words in the data base are distinct.

The next line contains the number of queries q ($1 \le q \le 10\,000$). The following q lines contain the user queries.

All the words in the input file consist of lowercase English letters The length of each word is between 1 and 10 characters.

Output

On the first line, print two integers. The first one must be the total number of keystrokes made by the user. The second one must be the total number of characters in all the queries plus the number of queries.

Examples

goondex.in	goondex.out
5 facemagaz 100 moogle 1000 izhstu 99 tco 50 narzan 5 5 facemagas tcopen izhstu izhmoloko mooglex	31 42
5 a 1000 aa 100 aaa 10 aaaa 1 b 15 8 aaaaa aaaa aaa aa a b ab bb	25 28

Problem C. Intersection

Input file:	`intersection.in`
Output file:	`intersection.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

This problem has not got a long tale. You have just to find the area of intersection of a convex polygon and a circle.

Input

The first line of input contains three integers: the coordinates of the center of the circle and its radius ($-10^4 \le X, Y \le 10^4$, $1 \le R \le 10^4$). The second line contains integer n, the number of vertices in the polygon ($3 \le n \le 40$). Next n lines contain coordinates of vertices of the polygon in clockwise or counter-clockwise order. It is guaranteed that no three points lie on one same line. The coordinates are integers and lie in the range from -10^4 to 10^4.

Output

Output one integer: the area of intersection of the polygon and the circle with absolute or relative error no more than 10^{-3}.

Examples

intersection.in	intersection.out

0 0 1
3
0 0
1 1
0 1
	0.392699

Problem D. LCA

Input file:	`lca.in`
Output file:	`lca.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Facepalm is a huge company with lots of employees. Each employee is either a developer or a manager. Developers and managers work in various teams. Each team has its own manager who is responsible for each person in the team. Additionally, each manager may be a part of another team which also has its own manager, and so on. The head manager is CEO Dark Uckerman, or just Duck.

For simplicity, we can assume that the company could be represented as a tree. Each developer is a leaf of the tree and all other vertices are managers. Each manager is responsible for all his subtree including himself.

Each employee has its own identifier. All identifiers are different integers and CEO has the identifier 1.

A common problem for two employees is to find a manager that is responsible for both of them. Naturally, the most convenient choice would be the "lowest" such manager, i. e., the one that is not responsible for another manager which in turn is also responsible for both employees. To make such a search simpler, *Facepalm* engineers have created a program which, given two employee identifiers, finds the identifier of their lowest common manager. Note that if employee A is responsible for employee B, their lowest common manager is A.

To test this program, the engineers had to run it for every pair of employees A and B such that identifier of A is not greater than identifier of B. All the output values were printed on the screen; value v was printed t_v times in total. An employee with identifier v is called a *duck star* if his identifier appeared on the screen most often, i. e., $t_v \geq t_u$ for all $1 \leq u \leq n$. Your task is to find the identifiers of all the duck stars working for *Facepalm*.

Input

The first line of input contains n, the total number of *Facepalm* employees ($1 \leq n \leq 10^5$). Each of the next $n-1$ lines contains a pair of identifiers a and b which means that either a is responsible for b or b is responsible for a; which of the above is true is not specified but easily deducible ($1 \leq a, b \leq n$, $a \neq b$). It is guaranteed that Duck, which has identifier 1, is responsible for all employees. It is also guaranteed that the given structure of the company is indeed a tree.

Output

On the first line, print the number of duck stars. On the second line, print their identifiers in any order. Each identifier must be printed exactly once.

Examples

lca.in	lca.out

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

Problem E. Tree picture

Input file:	`picture.in`
Output file:	`picture.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Two great companies *Facepalm* and *Perimeter* are planning to make a common huge project. This project will change the world, but now it is a really big secret.

It is clear that other companies have heard about it and they are really scared. So there are lots of spies that were sent to *Facepalm* and *Perimeter*. But the secret is so deep that all the spies have found nothing.

A few days ago, one of the spies has found a picture that seems to be connected with this project. There is a tree on this picture. And now we need to recognize that tree.

Input

On the first line of input, there are two integers: height n and width m of the picture ($1 \le n, m \le 20$). Each of the next n lines contains m characters. Each character is one of the following: '`O`' (ASCII code 79) is a vertex of the tree, '`+`' (ASCII code 43) is a place where where an edge changes direction, '`|`' (ASCII code 124) is a place where an edge goes vertically, '`-`' (ASCII code 45) is a place where an edge goes horizontally, '`.`' (ASCII code 46) denotes an empty space.

It is guaranteed that the picture is correct. Specifically, there is exactly one tree on the picture, there is no vertex adjacent to a '`+`' and there are no two adjacent '`+`' characters. Additionally, there is no edge which does not connect two vertices. The edges do not cross each other. Each '`+`' connects exactly two parts of an edge and edges always change its direction in '`+`'. Any two neighbour characters are connected to each other.

Output

You should output the tree that is shown on the picture.

On the first line print n, the number of vertices of the tree. On the next $n - 1$ lines, print the edges of the tree. Vertices must be numbered by integers from 1 to n. If there are several answers, you can output any of them.

Examples

picture.in	picture.out
`4 5`	`3`
`.....`	`1 2`
`+-O-+`	`1 3`
`|...|`	
`O...O`	
`5 5`	`7`
`O.O-O`	`1 2`
`|.|.|`	`2 3`
`O-O.O`	`2 4`
`|....`	`4 5`
`O....`	`5 6`
	`6 7`

Problem F. Planet

Input file:	`planet.in`
Output file:	`planet.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

All *Facepalm* applications work for our planet, Earth. And now this company wants to create the most important thing: an infinite screen that would protect the planet from asteroids.

Due to the fact that Bruce Willis does not work for *Facepalm*, the employees of the company have decided not to build the screen themselves, but use a new powerful technology instead. They created a satellite that will be situated near the Earth and shoot a laser ray. Then this laser shot can create a screen that looks like an infinite plane destroying everything that intersects it. Now, the major problem is to create an application which would not allow the plane to destroy the planet but touch it.

We can imagine Earth as a sphere in space and the satellite with a laser as a point and a ray from this point in space. You need to find any plane that will touch the sphere and contain all the points of the ray.

In this problem, you can assume that the line that contains the point and the ray does not touch or intersect the sphere.

Input

The first line of input contains four integers X, Y, Z and R: the coordinates of the center of the sphere and its radius ($-10^4 \le X, Y, Z \le 10^4$, $1 \le R \le 10^4$). The second line contains coordinates of the point p in space, integers x_p, y_p and z_p ($-10^4 \le x_p, y_p, z_p \le 10^4$). The third line contains coordinates of the vector whose direction is equal to the ray's direction. Its coordinates are integers x_v, y_v and z_v and at least one of them is non-zero ($-10^4 \le x_v, y_v, z_v \le 10^4$). So, the ray consists of points $(x_p + t \cdot x_v, y_p + t \cdot y_v, z_p + t \cdot z_v)$ for all $t \ge 0$.

Output

Output three numbers x_n, y_n and z_n: the coordinates of vector n orthogonal to the plane. Length of this vector should be equal to 1. Your answer will be considered correct if all distances differ from required ones by no more than 10^{-6}.

Examples

planet.in	planet.out

0 0 0 1
3 3 3
1 0 0
	0.000000000 -0.853850938 0.520517604

1 1 1 1
2 2 2
-1 0 0
	0.000000000 0.000000000 -1.000000000

Problem G. Poker

Input file:	`poker.in`
Output file:	`poker.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Is there anybody who doesn't like poker? The employees of *Facepalm* like it! Especially on a Friday night after work. Poker became very popular among some n developers from *Facepalm*. It is crowdy on each of their games. And the developers realized that it would be cool to have a special utility that can calculate the probability of winning for each of the players.

Let us recall the rules of Texas Holdem (it is a variant of poker which our programmers play). The players use a standard 52-card deck. Each card has a rank (ordered from 2 to A: 2, 3, ..., 9, T as ten, J as jack, Q as queen, K as king, A as ace). Additionally, each card has a suit which does not affect their ordering (S for spades, C for clubs, D for diamonds, H for hearts).

The game is played in four rounds. During the first round, each player takes two cards from the deck. During the second round, three cards are put face-up on the table so that every player can see them. During the third round, one more card is put face-up on the table and finally during the last, fourth round, one more card is put on the table. So, at the end there are five cards on the table face-up (they are named *community* cards) and each player has two *private* cards.

After that, players determine who has the better combination of cards. Each player in turn shows his two cards and chooses the best five-card combination from the set of seven cards formed of five community cards and his own two private cards. This five-card combination is called a *hand* and the two cards which are left do not count to the comparison. The winner is the player whose hand is stronger than other players' hands. In case no hand is stronger than all other hands, the game ends in a draw.

There are several hand categories and every hand from a stronger category is stronger than any combination from a weaker category. Two hands of the same category are compared using a tie-breaking rule specific for their category. Here the categories are listed from weakest to strongest:

- *High card* — Does not fit into any ranking below. When comparing with another High card hand, the ranks of the highest cards in the two hands are first compared. If there is a tie, the second highest cards in each hand are compared, and so on. (*Example*: QS, JH, 9C, 7H, 3D.)
- *One pair* — Two cards of the same rank. Pair with higher rank beats the lower pair. In case of a tie, the remaining three cards are used as tie-breakers, compared in the descending order of their ranks (as in High card). (*Example*: 6D, 6H, QD, 9H, 4S.)
- *Two pairs* — Two pairs of cards of the same rank. When comparing with another *Two pairs* hand, the higher pair is first compared, then the lower pair, and finally the rank of the fifth remaining card. (*Example*: JH, JS, TS, TD, 8S.)
- *Three of a kind* — Three cards of the same rank. Three-of-a-kind with the higher rank beats the lower one. In case of a tie, the remaining two cards are used as tie-breakers, compared in the descending order. (*Example*: 5S, 5H, 5D, JH, 6D.)
- *Straight* — Five cards in consecutive rank. An ace can either be accounted above a king or below a two, but not both, so wrapping is not allowed. Two straights are compared using the rank of the highest card (in the case of A, 2, 3, 4, 5, the highest card is considered to be 5). (*Example*: QH, JC, TH, 9D, 8D.)
- *Flush* — Five cards of the same suit. When comparing two *Flushes*, the rank of the highest card is first considered, then the second highest and so on (as in *High card*). (*Example*: AS, JS, 8S, 6S, 5S.)
- *Full house* — Three cards of the same rank, and two cards of same rank. When comparing with another *Full house*, the rank of the three cards is first compared, then the rank of the two cards. (*Example*: 7S, 7H, 7C, JC, JH.)
- *Four of a kind* — Four cards of the same rank. Two four-of-a-kinds are first compared by the ranks of the four cards. In case of a tie, the rank of the fifth card is used as a tie-breaker. (*Example*: 4C, 4D, 4H, 4S, TD.)

- *Straight flush* — A hand that is both a *Straight* and a *Flush*. Same tie-breaker as for a *Straight*. (*Example*: TH, 9H, 8H, 7H, 6H.)
- *Royal flush* — A special case of a *Straight flush* that ends with an ace. (*Example*: TS, JS, QS, KS, AS.)

You should calculate the probability to win the game for each player after each round. After the last round, you should print who won and what was his combination. For the purposes of this problem, in case of a draw, we assume that nobody wins.

Input

The first line of input contains an integer n, the number of players ($2 \le n \le 8$). The next n lines describe hands for each of the players. These lines each contain descriptions of two cards separated by one space. The card is described with two characters denoting its rank and suit, respectively. The ranks are denoted by '`2`', ..., '`9`', '`T`', '`J`', '`Q`', '`K`', and '`A`' (listed here in ascending order). The suits are denoted by '`C`', '`D`', '`H`', and '`S`'. The next line describes three cards: the cards on the table for the second round. The following two lines each describes one card: the next card on the table for the third and fourth rounds respectively. All cards are distinct.

Output

On the first line, print "`First bet`" without quotes. On the next n lines, print "`Player` i`:` A/B", where i is the number of player from 1 to n and A/B is the probability of winning for this player printed as an irreducible fraction after the first round.

On the next line, print "`Second bet`" without quotes. On the next n lines print "`Player` i`:` A/B", where i is the number of player from 1 to n and A/B is the probability of winning for this player printed as an irreducible fraction after the second round.

On the next line, print "`Third bet`". On the next n lines print "`Player` i`:` A/B", where i is the number of player from 1 to n and A/B is the probability of winning for this player printed as an irreducible fraction after the third round.

Finally, on the next line, print the result of the game. If the game ended in a draw, print "`No winner: Draw`" without quotes. Otherwise, print "`Winner is Player` i`,` s", where i is the number of player who wins (from 1 to n) and s is the name of the winning combination for this player. Combinations are: "`Royal flush`", "`Straight flush`", "`Four of a kind`", "`Full house`", "`Flush`", "`Straight`", "`Three of a kind`", "`Two pairs`", "`One pair`", "`Nothing`".

Examples

poker.in

```
2
2C 7S
JS QS
TS KS 6D
8S
AS
```

poker.out

```
First bet
Player 1: 250997/856152
Player 2: 596417/856152
Second bet
Player 1: 1/9
Player 2: 437/495
Third bet
Player 1: 0/1
Player 2: 1/1
Winner is Player 2, Royal flush
```

Problem H. Toys

Input file:	`toys.in`
Output file:	`toys.out`
Time limit:	6 seconds (*8 seconds for Java*)
Memory limit:	256 mebibytes

All boys love toys. Even grown-up engineers like toys! The toys just get more expensive.

One of the engineers of *Facepalm* has bought a birthday present to his kid. It is a set of colorful cubes. Each cube has its sides colored with one of six different colors. The cubes can be arranged in a row side-by-side. This line of cubes looks like a colorful snake with the leftmost cube being a head and the rightmost one being a tail.

The cubes can be placed in many different ways and the snake gets a new look each time but only colors of cube sides facing outwards matter. The kid really likes this toy and spends all his time laying out different snakes.

He uses the following operation to modify the snake. First, he chooses some consecutive segment of cubes from i-th to j-th position in the snake, inclusive, counting from the left (the snake head). Then he chooses one of six sides (front, back, top, bottom, left, right) and rotates that segment 90 degrees clockwise from the chosen side in such a way that the consecutive cubes in the segment remain connected by a face. If the set consisted from more than one cube and rotation was not from the right or left side, the snake becomes broken, so he rotates the segment once more in the same direction and then moves it (without any more rotations) to fill the gap in the snake and return it to the state of a single chain.

From time to time our engineer asks the kid how many colors are there on the upper sides of the cubes. Your task is to simulate all kids' actions and print the final configuration of the snake.

Input

The first line of input contains one integer n, the number of cubes ($1 \le n \le 10^5$). After that there are n lines, each describing one cube. The description contains six different integers from 1 to 6 in the order of *top*, *bottom*, *left*, *right*, *front*, *back* sides of the cube. These numbers are given without any separators. The next line contains the number of queries q ($1 \le q \le 10^5$). The following q lines describe the queries.

There are two types of queries.

The first one is denoted by a single capital English letter S followed by integer numbers i and j, $1 \le i \le j \le n$. This indicates a request to rotate a subset of cubes from i-th to j-th clockwise from the corresponding side: front if $S =$ '`F`', back if $S =$ '`B`', top if $S =$ '`U`', bottom if $S =$ '`D`', left if $S =$ '`L`' and right if $S =$ '`R`'. Consecutive tokens in this description are separated by spaces.

The second type of query is denoted by a single question mark ('`?`') which is a query for the number of occurrences of each of the six colors on the upper sides of the cubes after performing all rotations described in the previous lines of input.

Output

For each query for the colors occurrences, you should output six integers separated by spaces. Each of those integers is the number of cubes with top side of the corresponding color.

After all queries, you should output n lines. i-th of them should contain colors of i-th cube in the same format as in the input.

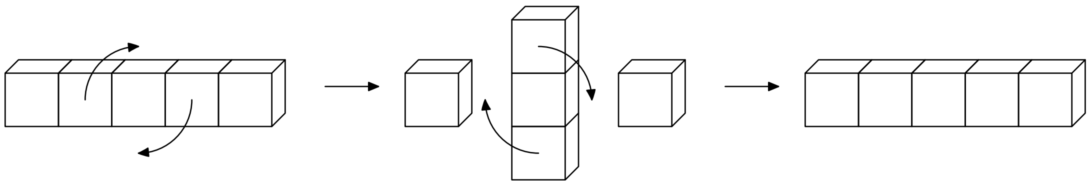
Examples

toys.in	toys.out

3
123456
123456
123456
6
?
F 1 1
?
U 2 3
?
R 1 2
	3 0 0 0 0 0
	2 0 1 0 0 0
	2 0 1 0 0 0
	562143
	654321
	124365

Note

Below is an example of a rotation “`F 2 4`” with $n = 5$.



Problem I. Union

Input file:	`union.in`
Output file:	`union.out`
Time limit:	2 seconds (*5 seconds for Java*)
Memory limit:	256 mebibytes

As you remember, in the problem "Tree picture" there was a story about a common huge project. This project is the union of two big companies. Now we finally can tell you some details about this union of the two big companies, *Facepalm* and *Perimeter*. It is not important what was in the previous problem. Now we want to speak about that tree. A tree is an undirected connected graph without cycles.

This tree is a model of something big and still secret. You don't need to know the details. All you have to know is that there is a tree with n vertices. All the edges of the tree have some weight. You will be given lots of queries. Each query asks you to find the number of edges on the path between two vertices which have the weight less or equal to some given threshold. Write a program to answer these queries.

Input

On the first line of input, there is an integer n, the number of vertices of the tree ($1 \le n \le 10^5$). Next $n-1$ lines describe the edges of the tree. Each line contains three integers a, b and w where a and b are the vertices that are connected by the current edge and w is the weight of this edge ($1 \le a, b \le n$, $a \ne b$, $1 \le w \le 10^6$). The next line contains the number of queries q ($1 \le q \le 10^5$). Each query consists of three integers v, u and k separated by spaces where v and u are the vertices that are the start and the end of the path and k is the threshold ($1 \le v, u \le n$, $1 \le k \le 10^6$).

Output

For each query, print one integer which is the number of edges on the given path which have the weight less than or equal to the given threshold.

Examples

union.in	union.out

3
1 2 1
1 3 2
3
1 2 2
2 3 2
2 3 1
	1
	2
	1

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

Problem J. WH.A.T.S.

Input file:	`whats.in`
Output file:	`whats.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

All the developers of *Facepalm* like the game "Failout 3". They spend a lot of time playing it, of course in their free time. As you know, "Failout" is famous for its battle system named WH.A.T.S. Using WH.A.T.S., the player can aim to individual parts of enemy body.

The developers like this battle system very much. They want to include it in their new game. Now the developers want to know the probability of the event that the player kills the enemy using WH.A.T.S.

The battle goes as follows. The player and the enemy take turns one after another, the player moves first. Initially, the player has HP_{player} hit points, and the enemy has HP_{enemy} hit points. There are T guns available to the player and P parts of enemy's body. For each gun and each body part of the enemy, you know the probability to hit it using that gun, and also you know the damage the enemy will take if that body part is hit with that gun. Additionally, for each gun you know WH.A.T.S. points VP_i which is the number of shots the player can make using the i-th gun during one turn.

On each turn, the player chooses gun number i and body part j and attacks the enemy VP_i times with that gun aiming in that part of the body. Each of VP_i shots inflicts $d_{i,j}$ damage with probability $p_{i,j}$ (and inflicts no damage with probability $1-p_{i,j}$). The outcomes of all shots are independent. If the player kills the enemy (makes HP_{enemy} less or equal to zero), the battle ends and the player wins. In the other case, the enemy will attack the player and inflict A units of damage with the probability P_{enemy}. After that, in case the player has zero or less hit points, the battle ends and the player loses. In the other case, there is player's turn again, and so on.

You can assume that the player uses the optimal strategy. Calculate the probability that the player wins.

Input

On the first line of input, there are two integers HP_{player} and HP_{enemy} ($1 \le \mathrm{HP}_{player}, \mathrm{HP}_{enemy} \le 100$). The next line contains two integers: the number of guns T and the number of enemy body parts P ($1 \le T, P \le 10$). Each of the next T lines contains the description of i-th gun. Each description starts with integer VP_i which is the WH.A.T.S. points of that gun ($1 \le \mathrm{VP}_i \le 5$). It is followed by P pairs of integers $p_{i,j}$ and $d_{i,j}$ which are the probability to hit the j-th part of the enemy body using the i-th gun (in percent) and the damage inflicted in case of a hit ($1 \le p_{i,j} \le 100$, $1 \le d_{i,j} \le 20$). The last line contains two integers P_{enemy} and A which are the probability of the enemy inflicting damage to the player (in percent) and the amount of damage in case of a hit ($1 \le P_{enemy} \le 100$, $1 \le A \le 100$).

Output

Output the probability that the player wins with absolute or relative error no more than 10^{-9}.

Examples

whats.in	whats.out

`100 20`
`2 2`
`2 50 10 70 5`
`1 10 20 100 1`
`100 1`
	`1.0000000000`

`1 1`
`1 1`
`1 50 1`
`50 1`
	`0.6666666667`