Problem A: Highway

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

The sparsely populated, rural province you live in does not have a lot of infrastructure, which – as more and more jobs move to the cities – causes the population increasingly more trouble. After years of careful consideration, the local government decided that a good way to tackle this problem would be to build a brand new highway between the only two notable cities in the area, without any exits in between.

As there's (physically) not much in the way of such a construction, they plan to build the highway along a straight line between the cities. They already drew said line in their planning map, but they still need to figure out exactly what parts of land they need to buy from the current owners, so as to not build anything on land they don't own, while also not buying land they don't intend to use.

Given the extent R to which the completed highway will deviate from the line (to each side), they need you to find a parametrization of the two lines parallel to it that describe the lateral boundaries of the planned highway and thus together enclose the area that will be covered by the construction.

Input

The inputs consists of four integers a, b, c and R, where the first three denote the parameters of the line $\ell = \{(x, y) \mid ax + by + c = 0\}$ they drew and the last one denotes the deviation to either side.

Output

Your output should consist of six numbers, the first three denoting the parameters of one deviated line, the other three those of the other one. All numbers should be printed with precision 10^{-6}.

Sample input and output

Input	Output
0 -1 1 1	0.00000000 -1.00000000 2.00000000 0.00000000 -1.00000000 0.00000000

Problem B: Interstellar objects

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

You just located two extremely fast moving interstellar objects on your 2-dimensional space map. Of course you now would like to know how close they will get to one another (since at their insane pace a collision would be really spectacular). However, since calculating this is pretty difficult, you decide to first compute how close the objects' trajectories (from the current point in time on) get to one another.

Input

The input consists of two lines, containing two pairs of integers each. The first pair denotes the coordinates of the current position of the respective object, the second pair those of another point your precalculations show to lie on its future trajectory (which is perfectly straight).

Output

Output one number – the distance between the objects' future trajectories – with precision 10^{-6}.

Sample input and output

Input	**Output**
0 0 1 0 0 1 0 2	1.000000

Problem C: Cutter

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

You have a paper strip that extends from 0 to w in y-direction and to $\pm$ infinity in x-direction. While you weren't watching, somebody got bored with a cutter in his hand and made lots of straight cuts on the strip. Now you want to find out whether the two ends are still connected.

Input

The input consists of

- one line containing N and w ($1 \leq N \leq 1000, 1 \leq w \leq 1000$) – the number of cuts and the width of the strip
- N lines describing the cuts, with the i-th line containing integers x_i, y_i, x'_i and y'_i ($-1000 \leq x_i, y_i, x'_i, y'_i \leq 1000$), denoting that the i-th cut is the straight line connecting (x_i, y_i) and (x'_i, y'_i).

Output

Output 1 if the strip is still connected, 0 otherwise.

Sample input and output

Input	**Output**
2 6 0 -1 2 4 -1 7 3 2	0
3 8 4 -1 -1 3 -3 -2 1 3 0 3 1 9	1

Problem D: Galactic warlords

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Will the galaxy see peace at last? All warlords have gathered to divide all of space between themselves. The negotiations have come quite far and the warlords have finally agreed on a peaceful way of deciding who gets what. The 2-dimensional galactic map must first be divided into sectors by splitting it along a set of infinite lines. The warlord with the largest battle fleet will choose one sector, then the warlord with the second largest fleet will choose some other sector and so on, until everyone has gotten a sector. This is then repeated until there are no sectors left.

Different sets of lines have been suggested, and it is up to you to present these alternatives to the meeting. To make sure that there will be peace, you are ready to modify the suggestions slightly. You have some experience with warlords and know that no warlord will settle for less space than anyone else, so for there to be peace, all of them must get the exact same area on the map. Since space is infinite, so is the map. Some sectors will therefore have infinite area, so that is the amount of space everyone will want. How many extra lines will you have to add to make sure each warlord can get at least one sector with infinite area?

Input

The input consists of

- one line containing W and N ($1 \leq W, N \leq 100$) – the number of warlords and the number of lines in the suggested division of space
- N lines each containing integers x_1, y_1, x_2 and y_2 ($-10^4 \leq x_i, y_i \leq 10^4$), denoting a line that is passing through the points (x_1, y_1) and (x_2, y_2) on the galactic map. It is guaranteed that $(x_1, y_1) \neq (x_2, y_2)$ for all given lines.

Output

Output the number of lines you will have to add to this suggestion to satisfy all warlords.

Sample input and output

Input	Output
2 1 1 1 -2 0	0
5 3 0 5 5 5 0 0 1 1 2 2 3 3	1

Problem E: Mischievous Monkeys

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Little Henry just wanted to spend the afternoon playing in the park – now he is devastated: He had just gotten there and was looking for other kids to play with when some mischievous capuchin monkeys stole his beloved ball and, after passing it around for a while, left it on a water lily leaf in the middle of a large, rectangular pond.

Of course now he desperately wants to get it back, however, he faces some difficulties: First of all, he can't swim. Thus the only way for him to reach the ball is by using the pond's many water lilies (just like the monkeys did). Secondly, while all of their leafs are large enough to carry his weight, of course the lilies don't cover the entire surface of the pond – most of the time, he will have to jump between them to get ahead. Thirdly, he can only cover a certain distance d with each jump. Fourthly, Henry suspects he won't be able to jump arbitrarily often, so he wants to minimize the number of jumps he needs to make to get to his ball. Can you help him figure out which path to take?

Input

The input consists of

- one line containing integers a and b ($1 \leq a, b \leq 10^4$) – denoting that the pond covers the area $[0, a] \times [0, b]$
- one line containing an integer n ($1 \leq n \leq 1000$) – the number of water lily leafs – Henry's ball lies on the n-th leaf
- n lines describing the leafs, with each of them containing three integers M_x, M_y and r ($0 < M_x - r < M_x + r < a, 0 < M_y - r < M_y + r < b$), denoting that the respective leaf is a circle with center $M = (M_x, M_y)^T$ and radius r
- one line containing an integer d ($1 \leq d \leq \min(a, b)$) – the distance Henry can cover with a single jump.

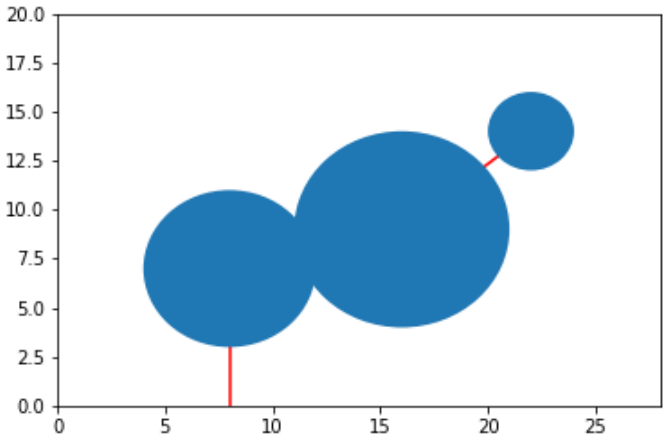
Output

Output the minimum number of jumps Henry needs to reach his ball. He can start from any point on the pond's boundary. If he can't reach the ball at all, output ":(".

Sample input and output

Input	Output
28 20 3 8 7 4 16 9 5 22 14 2 3	2
10 10 1 5 5 3 1	:(

Visualization of the first sample:



Problem F: Mirrors

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Given a list of mirrors (reflecting on both sides), you want to find out if there is one among them in which you can see yourself directly. You are a point. You can only see yourself if there is some $\varepsilon > 0$, such that you can see an area of ε around yourself in both directions (so you can't see yourself on the edge of a mirror). Mirrors may not intersect each other.

Input

The input consists of

- one line containing N ($1 \leq N \leq 100$) – the number of mirrors
- one line containing x and y ($-1000 \leq x, y \leq 1000$) – the coordinates of your position
- N lines describing the mirrors, with the i-th line containing integers x_i, y_i, x'_i, y'_i ($-1000 \leq x_i, y_i, x'_i, y'_i \leq 1000$), denoting that the i-th mirror is described by the segment connecting (x_i, y_i) and (x'_i, y'_i).

Output

Output 1, if there is a mirror in which you can see yourself, 0 otherwise.

Sample input and output

Input	Output
2 0 0 3 1 1 3 3 -1 3 -3	1
2 0 0 4 1 4 -1 3 1 1 -1	0