Problem A. Amsterdam Distance

Source file name:	amsterdam.c, amsterdam.cpp, amsterdam.java, amsterdam.py
Input:	`Standard`
Output:	`Standard`

Your friend from Manhattan is visiting you in Amsterdam. Because she can only stay for a short while, she wants to see as many tourist attractions in Amsterdam in as little time as possible. To do that, she needs to be able to figure out how long it takes her to walk from one landmark to another. In her hometown, that is easy: to walk from point $m = (m_x, m_y)$ to point $n = (n_x, n_y)$ in Manhattan you need to walk a distance

$$|n_x - m_x| + |n_y - m_y|, \tag{1}$$

since Manhattan looks like a rectangular grid of city blocks. However, Amsterdam is not well approximated by a rectangular grid. Therefore, you have taken it upon yourself to figure out the shortest distances between attractions in Amsterdam. With its canals, Amsterdam looks much more like a half-disc, with streets radiating at regular angles from the center, and with canals running the arc of the circle at equally spaced intervals. A street corner is given by the intersection of a circular canal and a street which radiates from the city center.

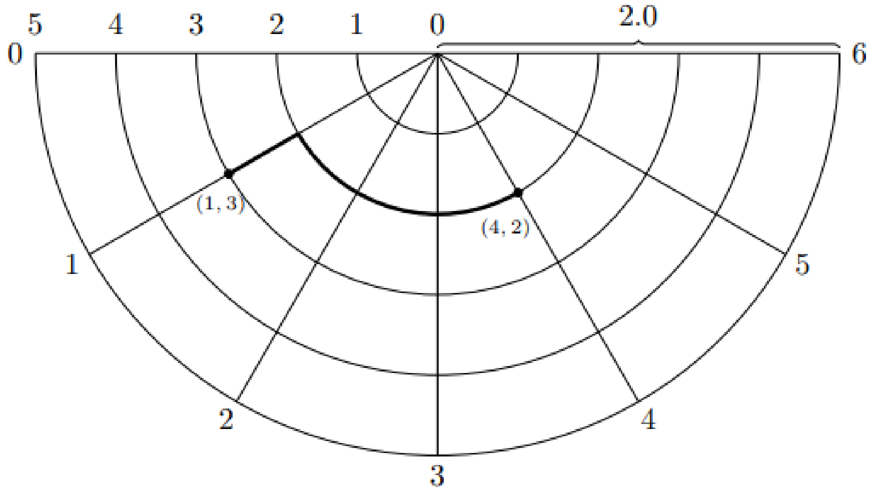


Figure 1: The first sample input.

Depending on how accurately you want to model the street plan of Amsterdam, you can split the city into more or fewer half rings, and into more or fewer segments. Also, to avoid conversion problems, you want your program to work with any unit, given as the radius of the half circle. Can you help your friend by writing a program which computes the distance between any two street corners in Amsterdam, for a particular approximation?

Input

The input consist of

- One line with two integers M, N and a floating point number R.
  - $1 \leq M \leq 100$ is the number of segments (or 'pie slices') the model of the city is split into.
  - $1 \leq N \leq 100$ is the number of half rings the model of the city is split into.

  - $1 \leq R \leq 1000$ is the radius of the city.

- One line with four integers, a_x, a_y, b_x, b_y, with $0 \leq a_x, b_x \leq M$, and $0 \leq a_y, b_y \leq N$, the coordinates of two corners in the model of Amsterdam

Output

Output a single line containing a single floating point number, the least distance needed to travel from point a to point b following only the streets in the model. The result should have an absolute or relative error of at most 10^{-6}.

Example

Input	Output

`6 5 2.0`
`1 3 4 2`
	`1.65663706143592`

`9 7 3.0`
`1 5 9 5`
	`4.28571428571429`

`10 10 1.0`
`2 0 6 0`
	`0`

Problem B. Bearly Made It

Source file name:	bearly.c, bearly.cpp, bearly.java, bearly.py
Input:	`Standard`
Output:	`Standard`

Barney the polar has wandered off on an adventure. Lost in thought, he suddenly realizes he has strayed too far from his mother and is stuck on an ice shelf. He can still see her in the distance, but the only way back is by crossing a group of other ice shelves, all of which are perfectly circular. He is very scared, and can not swim. Barney's mother, getting a little tired of her son's shenanigans, decides to wait and let him figure this out for himself. Can you help Barney get home? He is in a hurry.

Input

- The first line of input contains four integers, $-10^6 \leq x_b, y_b, x_m, y_m \leq 10^6$, where (x_b, y_b) is Barney's location and (x_m, y_m) is the location where Barney's mom is waiting.
- The next line contains a single integer $1 \leq n \leq 25$, the number of ice shelves.
- After this n lines follow. Each line holds three integers: $-10^6 \leq x_i, y_i \leq 10^6$ and $1 \leq r_i \leq 10^6$, the coordinates of the center of the shelf and its radius. A shelf consists of all points at distance r_i or less to (x_i, y_i).

Both bears are on a shelf at the start of Barney's journey home. Shelves can both touch and overlap.

Output

The minimal distance Barney has to travel to be reunited with his mother. The result should have a relative error of at most 10^{-6}.

If there is no way for Barney to make it home, output "`impossible`". (Do not worry about Barney's well-being in this scenario. His mother will swim out to save him.)

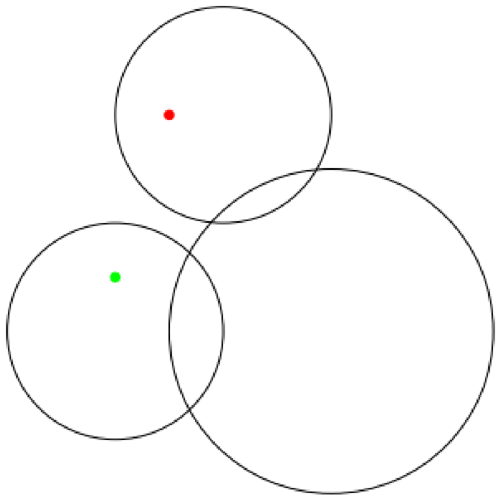


Figure 2: Illustration of the third example input.

Example

Input	Output

0 0 6 0
2
1 1 2
5 1 2
	6.32455532034

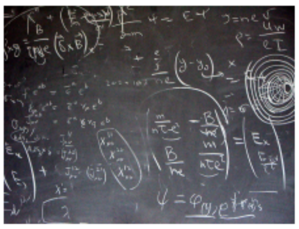
0 0 7 0
2
1 1 2
6 1 2
	impossible

0 0 1 3
3
0 -1 2
4 -1 3
2 3 2
	4.269334912857045697

Problem C. Collatz Conjecture

Source file name:	collatz.c, collatz.cpp, collatz.java, collatz.py
Input:	`Standard`
Output:	`Standard`

In 1978 AD the great Sir Isaac Newton, whilst proving that `P` is a strict superset of `NP`, defined the Beta Alpha Pi Zeta function `f` as follows over any sequence of positive integers $a_1, \ldots, a_n$. Given integers $1 \leq i \leq j \leq n$, we define $f(i, j)$ as $gcd(a_i, a_{i+1}, \ldots, a_{j-1}, a_j)$.


Picture by mscolly via Flickr.

About a century later Lothar Collatz applied this function to the sequence $1, 1, 1, \ldots, 1$, and observed that f always equalled 1. Based on this, he conjectured that f is always a constant function, no matter what the sequence a_i is. This conjecture, now widely known as the Collatz Conjecture, is one of the major open problems in botanical studies. (The Strong Collatz Conjecture claims that however many values f takes on, the real part is always $\frac{1}{2}$.)

You, a budding young cultural anthropologist, have decided to disprove this conjecture. Given a sequence a_i, calculate how many different values f takes on.

Input

The input consists of two lines.

- A single integer $1 \leq n \leq 5 \cdot 10^5$, the length of the sequence.
- The sequence $a_1, a_2, \ldots, a_n$. It is given that $1 \leq a_i \leq 10^{18}$.

Output

Output a single line containing a single integer, the number of distinct values f takes on over the given sequence.

Example

Input	Output
4 9 6 2 4	6
4 9 6 3 4	5

Problem D. Easter Eggs

Source file name:	easter.c, easter.cpp, easter.java, easter.py
Input:	`Standard`
Output:	`Standard`

Easter is coming and the Easter Bunny decided to organise a chocolate egg hunt for the children. He will hide two types of eggs: blue milk chocolate and red dark chocolate. In the field there are some redberry and some blueberry plants where the Easter Bunny could hide the eggs. Red eggs should be hidden in a redberry plant and blue eggs in a blueberry plant.

Picture by wackystuff via Flickr.

The local government has issued a permit for the event, under the condition that exactly N eggs are hidden. As they do not pay for the dental care plans of the local children, the Easter Bunny gets to decide himself how many eggs to hide of each colour.

According to the yearly tradition, there is a big reward for the first child to find both a red and a blue egg. In order to make the hunt as challenging as possible, the Easter Bunny wants to maximise the minimum distance between a red and a blue egg. To keep things fair, he will hide at most one egg in each plant. Your task is to write a program to help him accomplish his goal.

Input

The input consists of the following:

- one line containing three integers N, B, R, the number of eggs to hide $N \leq 250$, the number of blueberry plants $B < N$ and the number of redberry plants $R < N$;
- B lines, each containing two integers $-10^4 \leq x, y \leq 10^4$, indicating the coordinates (x, y) of a blueberry plant;
- R lines, each containing two integers $-10^4 \leq x, y \leq 10^4$, indicating the coordinates (x, y) of a redberry plant.

The $B + R$ plants are guaranteed to have distinct coordinates. Moreover, N is guaranteed to satisfy $N \leq B + R$.

Output

Output a single line containing a floating point number, D, the largest minimum distance between a red and a blue egg that can be achieved. You are required to output D with absolute precision 10^{-6}, i.e. with at least 6 decimal places.

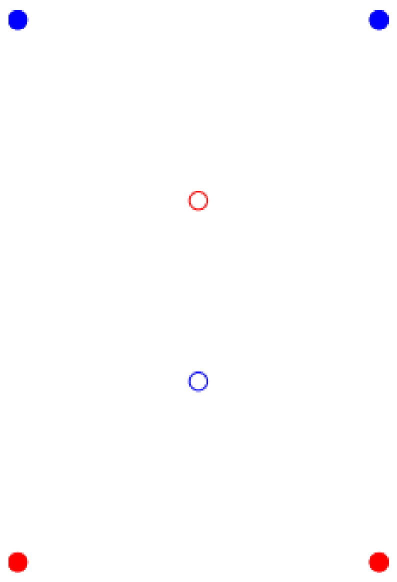


Illustration of the second example input. The eggs are hidden in the four filled bushes.

Example

Input	Output

3 2 2
0 0
1 0
2 0
3 0
	2.000000000000000

4 3 3
0 0
1 2
-1 2
0 1
-1 -1
1 -1
	3.000000000000000

Problem E. Falling Apart

Source file name:	falling.c, falling.cpp, falling.java, falling.py
Input:	`Standard`
Output:	`Standard`

After acquiring a new integer and showing it off to other couples at a cocktail party, Alice and Bob headed home for a good night of sleep. As their integer was quite large, they were forced to carry it together. Then, on the Boole Boulevard, right by the Bayes Bay, disaster struck. History does not tell us which of our two protagonists stumbled first, but stumble they did, and their integer shattered into n positive integral pieces on the pavement in front of them.

Picture by gratuit via Freeimageslive.

The couple's marriage, already under financial stress due to the expensive integer acquisition, did not survive this event, and Bob and Alice resolved to separate. Thus the question was raised on how to split the remnants of their integer. Bob and Alice decided to play a game with the remaining n pieces: the two would repeatedly choose pieces in alternating turns until none were left.

Bob and Alice, both very materialistic, seek to acquire the largest possible sum of integers possible. Compute the value of the integers each of them ends up with. Assume both players play optimally. Since A comes before B in the alphabet, Alice moves first.

Input

The input consists of two lines.

- A single integer $1 \leq n \leq 15$, the number of pieces.
- The values of the pieces $a_0, a_1, \ldots, a_{n-1}$, space-separated. It is given that $1 \leq a_i \leq 100$.

Output

Output a single line containing two integers, the combined value of Alice's pieces, and the combined value of Bob's pieces.

Example

Input	Output

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

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

Problem F. Going Dutch

Source file name:	dutch.c, dutch.cpp, dutch.java, dutch.py
Input:	Standard
Output:	Standard

You and your friends have just returned from a beautiful vacation in the mountains of the Netherlands. When on vacation, it's annoying to split the bill on every expense every time, so you just kept all the receipts from the vacation, and wrote down who paid how much for who. Now, it is time to settle the bill.

Picture by ben_osteen on Flickr.

You could each take all the receipts showing that someone paid something for you, and then pay that person back. But then you would need a *lot* of transactions, and you want to keep up the lazy spirit from your trip. In the end, it does not matter who transfers money to whom; as long as in the end, everyone's balance is 0.

Can you figure out the least number of transactions needed to settle the score? Assume everyone has enough spare cash to transfer an arbitrary amount of money to another person.

Input

Input consists of

- A line containing two integers M, the number of people in the group, with $1 \leq M \leq 20$, and N, the number of receipts from the trip, with $0 \leq N \leq 1000$.
- N lines, each with three integers a, b, p, where $0 \leq a, b < M$, and $1 \leq p \leq 1000$, signifying a receipt showing that person a paid p euros for person b.

Output

Output a single line containing a single integer, the least number of transactions necessary to settle all bills.

Example

Input	Output

4 2
0 1 1
2 3 1
	2

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

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

Problem G. Hoarse Horses

Source file name:	hoarse.c, hoarse.cpp, hoarse.java, hoarse.py
Input:	`Standard`
Output:	`Standard`

Farmer Bob's horses all are hoarse and may have a cold - even the pony is a little hoarse! Because of that, all the farm animals need to be individually quarantined. To separate the animals, Farmer Bob has a set of n fences that cannot be crossed. Unfortunately, Farmer Alice has taken all of Farmer Bob's fences and placed them arbitrarily in the plane! Farmer Bob has no time to rearrange these fences - he must leave them as is.

Picture by Alistair Hamilton via Flickr.

Help Farmer Bob calculate how many of his prized barnyard animals he can quarantine. That is, Farmer Bob wants to place as many animals inside non-empty regions enclosed by the fences, such that no animal can reach another animal, and no animal can escape to infinity.

Each fence is given by a line segment between two points. It is given that no three fences go through the same point. Fences are allowed to cross each other.

Input

- A single integer $1 \leq n \leq 1000$, the number of fences.
- Then n lines follow, each with four integers $-10^9 \leq x_1, y_1, x_2, y_2 \leq 10^9$. These are the endpoints of the fences, each fence is given by a straight line segment between two endpoints.

Output

Output a single line containing a single integer c, the maximum number of animals Farmer Bob can quarantine.

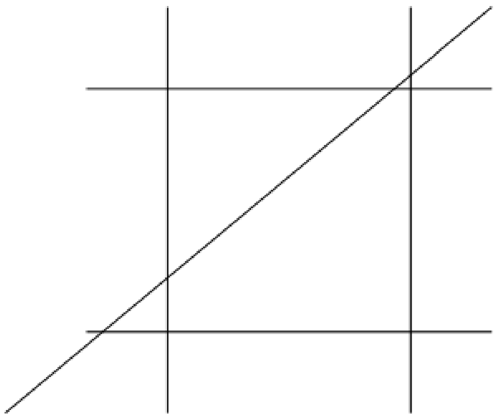


Figure 4: Illustration of the third example input.

Example

Input	Output
1	0
1 1 1 2	
4	1
1 0 1 5	
4 0 4 5	
0 1 5 1	
0 4 5 4	
5	4
1 0 1 5	
4 0 4 5	
0 1 5 1	
0 4 5 4	
-1 0 5 5	

Problem H. Irrational Division

Source file name:	irrational.c, irrational.cpp, irrational.java, irrational.py
Input:	`Standard`
Output:	`Standard`

Your family has been blessed with chocolate! A huge piece of chocolate has been given to you and your sister to share. However, as you gobbled up the large majority last time, your parents have invented a game to keep things fair (and to keep you occupied while they hide all the other chocolate). To keep things interesting, they have given you a rectangular piece of chocolate, which consists of little squares of both dark chocolate and white chocolate in a chessboard pattern. While you and your sister both love dark chocolate, you hate white chocolate! So, both you and your sister want as much dark chocolate as possible, while simultaneously obtaining as little white chocolate as possible. Every dark piece of chocolate you obtain gives you 1 meaningless unit of happiness, while a white piece lowers your happiness by 1 meaningless unit (and the same holds for your sister). Now, while you love your sister very much, there is always heavy competition between siblings, so your goal is to maximize the difference of your obtained happiness and her obtained happiness (while she tries to do the opposite, obviously).

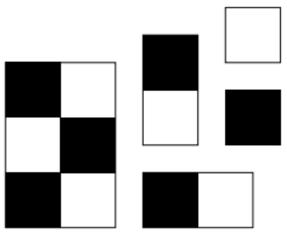


Figure 5: The example game described in the problem statement.

The game works as follows. Your parents place a $p \times q$-rectangle of the aforementioned mixed chocolate on a table. You are situated on the west side of the table and your sister on the south side. The side of length p is parallel to the north-south line, while the side of length q is parallel to the east-west line. Furthermore, the north-west square is made of dark chocolate. Then, starting with yourself, you take turns breaking off blocks of chocolate (which you can keep). You can break off any positive number of entire columns from the west side, while your sister breaks off any positive number of entire rows from the south side. You repeat this process until no more chocolate is left. Your sister is very smart and will always play the game perfectly.

A game might proceed like this, for example: you and your sister start with a 3×4-rectangle. You decide to break off 2 columns, obtaining 3 dark and 3 white chocolate squares, netting a happiness of zero. Your sister then breaks off 1 row, obtaining 1 dark and 1 white squares as well, so no happiness for her either. You then take a single column, which nets you nothing again, after which your sister decides to break off one row, which nets her 1 happiness! You then take the last piece, which makes you lose a unit of happiness, so your total score is $-1 - 1 = -2$. See the figure. (Note: the strategies used here might not be optimal.)

Input

Given are two positive integers p and q, both at most 100, the height and width of the chocolate rectangle.

Output

Output the largest possible difference (in your favour) between your net happiness and your sister's net happiness.

Example

Input	Output
`1 2`	`2`
`2 2`	`0`
`4 3`	`0`

Problem I. Jumping Choreography

Source file name:	jumping.c, jumping.cpp, jumping.java, jumping.py
Input:	`Standard`
Output:	`Standard`

As you may know, the frogs are the biggest show-offs of the entire animal kingdom. Some weeks ago, they greatly impressed the other animals by forming a large tower. However, the frog king wants to surpass this performance at the next Benelux Amphibian Pillaring Ceremony (BAPC). He wants the frogs to perform a difficult dance which will end in a climax where all frogs form a tower. You have been appointed choreographer and will practice with the frogs in the following months.

A frog dance is a form of line dance: a certain number of frogs line up and then perform a sequence of jumps, where every jump is either to the left or the right. The frog king decided to make this particular dance more interesting. Firstly, he dictated that the frogs have to increase the length of each jump. This means that for any frog, its first jump will be of length 1, the second of length 2, the third of length 3, and so on. Secondly, the dance should end with all frogs on one big tower. Thirdly, the total number of jumps that the frogs make should be as low as possible, in order to make the dance flashy and impressive-looking.

Since the king is a perfectionist, he wants the dance to be flawless. He has provided you with a team of excellent frog dancers, their starting positions, and the place he wants the frogs to form a tower at the end of the dance. However, the king still isn't convinced that the dance will be as perfect as he wants it to be, so he visits the rehearsal *every day* in order to make a change: he might find another frog that is very good at dancing and add it to the line-up, or he might feel that a frog is not good enough and remove him/her. He can even change the position of the final tower if he feels like it.

At the end of every day, the frog king wants to see the dance performed in the most efficient way possible, i.e. with the lowest total number of jumps.

Input

- A single line containing two integers $0 \leq n \leq 5000$ and $0 \leq t \leq 10^6$, the initial number of frogs and the initial position of the frog tower.
- The second line contains n integers $0 \leq p_i \leq 10^6$, the starting positions of these frogs.
- Then follows a line with an integer $0 \leq C \leq 10^6$, the number of changes the king makes.
- C lines follow, each of one of the following three forms.
  - A line of the form $+$ a indicates that the king adds a frog at position a.
  - A line of the form $-$ a indicates that the king removes a frog from position a. You may assume that at least one frog started from this position before removing it.
  - A line of the form t a indicates that the king changes the position of the frog tower to a.

In each case a is between 0 and 10^6 inclusive. **It is guaranteed that the number of times the kings adds or removes a frog is at most 5000**.

Output

For each of the C modifications, print one line containing the lowest total number of jumps of the dance after applying the modification

Example

Input	Output
1 1	0
0	1
7	3
t 0	2
t 1	3
t 2	5
t 3	3
t 4	
t 5	
t 6	
3 0	11
2 6 6	6
10	5
t 1	9
t 2	4
t 3	3
t 4	7
t 5	9
t 6	9
t 7	10
t 8	
t 9	
t 10	
0 0	0
	1
7	1
t 3	2
+ 4	6
t 5	3
+ 6	5
t 2	
- 4	
t 1	

Problem J. Lemonade Trade

Source file name:	lemonade.c, lemonade.cpp, lemonade.java, lemonade.py
Input:	`Standard`
Output:	`Standard`

The lunch break just started! Your mother gave you one litre of pink lemonade. You do not like pink lemonade and want blue lemonade instead. Fortunately for you the other children in class are willing to trade with you.

Picture by David Wargert via Flickr.

Each of them is willing to offer you any quantity of the virtually infinite amount of lemonade they got from their mother, in exchange for their favourite lemonade, according to some exchange rate. The other children are sitting in a long row in the class room and you will walk along the row, passing each child only once. You are not allowed to walk back! Of course, you want to maximise the amount of blue lemonade you end up with. In case you can obtain more than 10 litres of blue lemonade, this is more than you will need, and you will throw away any excess (and keep the 10 litres).

Fortunately, you know in advance what everybody is offering for trade. Your task is to write a program to find the maximum amount of blue lemonade that you can obtain.

Input

The input consists of the following:

- One line containing a single integer $0 \le N \le 10^5$, the number of children in the class room, excluding yourself;
- N lines, each containing two strings O, W and a floating point number $0.5 < R < 2$, the name of the lemonade that is offered, the name of the lemonade that is wanted, and the exchange rate: for every litre of lemonade W that you trade you get R litres of lemonade O in return.

All strings are guaranteed to have at most 10 alphanumeric characters.

Output

Output a single line containing a single floating point number M, the maximum amount (in litres) of blue lemonade that you can obtain. In case you could obtain more than 10 litres, M is considered to be 10. You are required to output M with absolute precision 10^{-6}.

Example

Input	Output
3 blue pink 1.0 red pink 1.5 blue red 1.0	1.500000000000000
2 blue red 1.0 red pink 1.5	0.000000000000000
4 orange pink 1.9 yellow orange 1.9 green yellow 1.9 blue green 1.9	10.000000000000000
8 red pink 1.9 orange red 1.9 yellow orange 1.9 green yellow 1.9 indigo green 0.6 violet indigo 0.6 purple violet 0.6 blue purple 0.6	1.688960160000000

Problem K. Manhattan Mornings

Source file name:	manhattan.c, manhattan.cpp, manhattan.java, manhattan.py
Input:	`Standard`
Output:	`Standard`

As a New Yorker you are always very busy. Apart from your long work day you tend to have a very long list of errands that need to be done on any particular day. You really hate getting up early so you always end up going over your to-do list after work, but this is starting to seriously hurt your free time.

One day you realize that some of the places you have to go by lie on your walk to the office, so you can visit them before work. The next day you notice that if you take a slightly different route to work you can run most of your errands without increasing the length of your route. Since errands take a negligible amount of time, you don't even have to get up any earlier to do this! This nice little side effect of the grid-like New York streets gets you thinking. Given all the locations of your errands on the New York grid, how many can you visit on your way to work without getting up any earlier?

The New York grid is modelled with streets that are parallel to the x-axis and avenues that are parallel to the y-axis. Specifically, there is a street given by $y = a$ for every $a \in \mathbb{Z}$, and there is an avenue given by $x = b$ for every $b \in \mathbb{Z}$. It is given that an errand always takes place on an intersection between a street and an avenue. Since you walk to your work, you can use all roads in both directions.

Input

- The first line contains an integer $0 \leq n \leq 10^5$, the number of errands you have to run that day.
- The next line contains four integers $0 \leq x_h, y_h, x_w, y_w \leq 10^9$ corresponding to the locations of your house and workplace.
- The next n lines each contain two integers $0 \leq x_i, y_i \leq 10^9$, the coordinates of your i_{th} errand.

Output

Output a single line, containing the number of errands you can run before work without taking a longer route than necessary.

Example

Input	Output
3	2
0 0 6 6	
5 4	
2 6	
3 1	
5	3
2 1 0 0	
0 0	
0 1	
2 0	
2 1	
3 1	
4	2
200 100 100 200	
50 150	
200 200	
100 100	
100 100	