Problem A. Aerodynamics

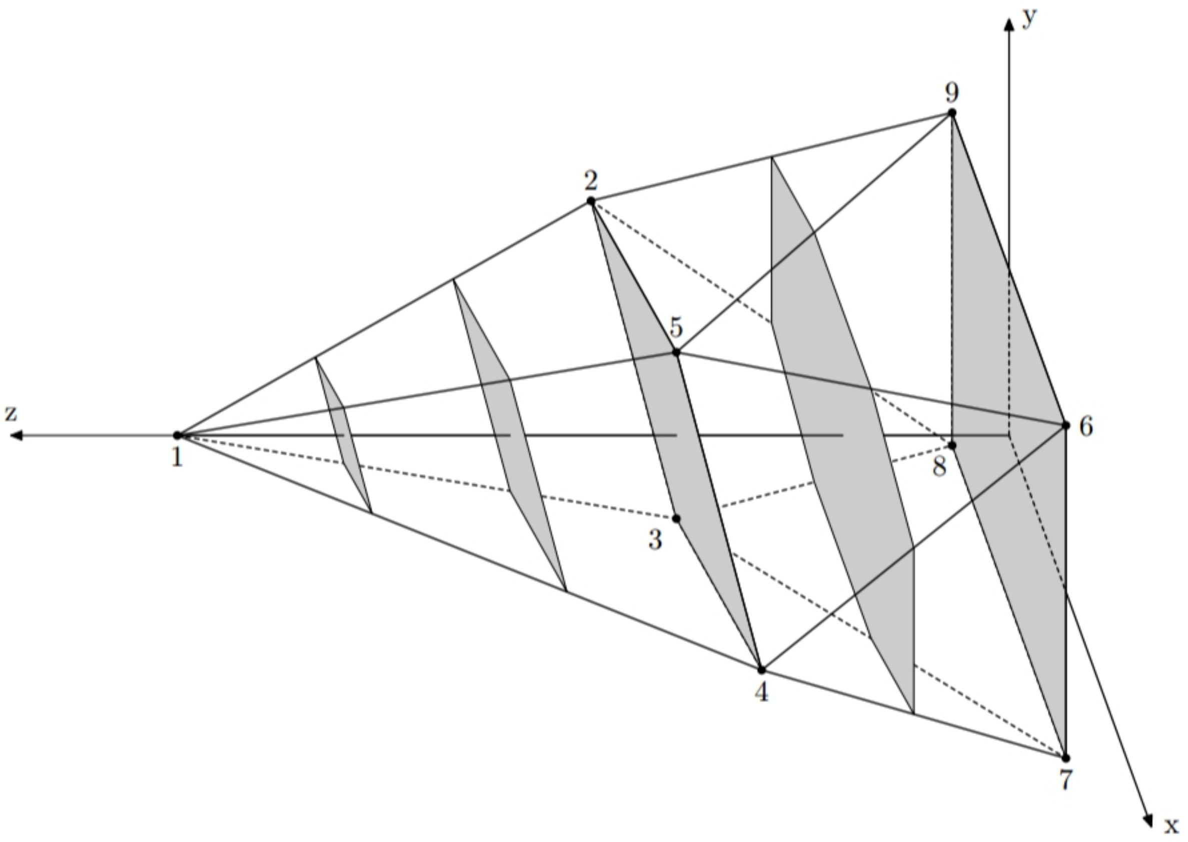
Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 mebibytes

Bill is working in a secret laboratory. He is developing missiles for national security projects. Bill is the head of the aerodynamics department.

One surprising fact of aerodynamics is called Whitcomb area rule. An object flying at high-subsonic speeds develops local supersonic airflows and the resulting shock waves create the effect called wave drag. Wave drag does not depend on the exact form of the object, but rather on its *cross-sectional profile*.

Consider a coordinate system with OZ axis pointing in the direction of object's motion. Denote the area of a section of the object by a plane $z = z_0$ as $S(z_0)$. Cross-sectional profile of the object is a function S that maps z_0 to $S(z_0)$. There is a perfect aerodynamic shape called Sears-Haack body. The closer cross-sectional profile of an object to the cross-sectional profile of Sears-Haack body, the less wave drag it introduces. That is an essence of Whitcomb area rule.

Bill's department makes a lot of computer simulations to study missile's aerodynamic properties before it is even built. To approximate missile's cross-sectional profile one takes samples of $S(z_0)$ for integer arguments z_0 from z_{min} to z_{max}.



Your task is to find the area $S(z_0)$ for each integer z_0 from z_{min} to z_{max}, inclusive, given the description of the missile. The description of the missile is given to you as a set of points. The missile is the minimal convex solid containing all the given points. It is guaranteed that there are four points that do not belong to the same plane.

Input

The first line of the input file contains three integer numbers: n, z_{min} and z_{max} ($4 \leq n \leq 100$, $0 \leq z_{min} \leq$

$z_{max} \leq 100$). The following n lines contain three integer numbers each: x, y, and z coordinates of the given points. All coordinates do not exceed 100 by their absolute values. No two points coincide. There are four points that do not belong to the same plane.

Output

For each integer z_0 from z_{min} to z_{max}, inclusive, output one floating point number: the area $S(z_0)$. The area must be precise to at least 5 digits after decimal point.

Example

standard input	standard output

9 0 5
0 0 5
-3 0 2
0 -1 2
3 0 2
0 1 2
2 2 0
2 -2 0
-2 -2 0
-2 2 0
	16.00000
	14.92000
	10.08000
	4.48000
	1.12000
	0.00000

Problem B. Asteroids

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Association of Collision Management (ACM) is planning to perform the controlled collision of two asteroids. The asteroids will be slowly brought together and collided at negligible speed. ACM expects asteroids to get attached to each other and form a stable object.

Each asteroid has the form of a convex polyhedron. To increase the chances of success of the experiment ACM wants to bring asteroids together in such manner that their centers of mass are as close as possible. To achieve this, ACM operators can rotate the asteroids and move them independently before bringing them together.

Help ACM to find out what minimal distance between centers of mass can be achieved.

For the purpose of calculating center of mass both asteroids are considered to have constant density.

Input

Input file contains two descriptions of convex polyhedra.

The first line of each description contains integer number n — the number of vertices of the polyhedron ($4 \le n \le 60$). The following n lines contain three integer numbers x_i, y_i, z_i each — the coordinates of the polyhedron vertices ($-10^4 \le x_i, y_i, z_i \le 10^4$). It is guaranteed that the given points are vertices of a convex polyhedron, in particular no point belongs to the convex hull of other points. Each polyhedron is non-degenerate.

The two given polyhedra have no common points.

Output

Output one floating point number — the minimal distance between centers of mass of the asteroids that can be achieved. Your answer must be accurate up to 10^{-5}.

Example

standard input	standard output

8
0 0 0
0 0 1
0 1 0
0 1 1
1 0 0
1 0 1
1 1 0
1 1 1
5
0 0 5
1 0 6
-1 0 6
0 1 6
0 -1 6
	0.75

Problem C. Driving Directions

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

Contrary to the popular belief, alien flying saucers cannot fly arbitrarily around our planet Earth. Their touch down and take off maneuvers are extremely energy consuming, so they carefully plan their mission to Earth to touch down in one particular place, then hover above the ground carrying out their mission, then take off. It was all so easy when human civilization was in its infancy, since flying saucers can hover above all the trees and building, and their shortest path from one mission point to the other was usually a simple straight line — the most efficient way to travel. However, modern cities have so tall skyscrapers that flying saucers cannot hover above them and the task of navigating modern city became quite a complex one. You were hired by an alien spy to write a piece of software that will ultimately give flying saucers driving directions throughout the city. As your first assignment (to prove your worth to your alien masters) you should write a program that computes the shortest distance for a flying saucer from one point to another. This program will be used by aliens as an aid in planning of mission energy requirements.

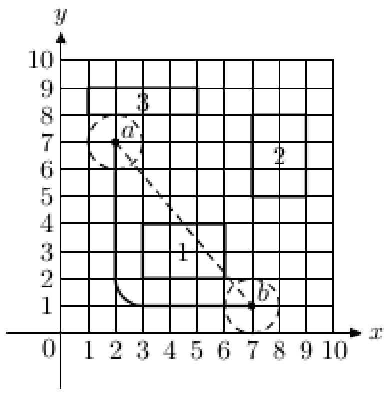
The problem is simplified by several facts. First of all, since flying saucer can hover above most of the buildings, you are only concerned with locations of skyscrapers. Second, the problem is actually two-dimensional — you can look at everything "from above" and pretend that all objects are situated on OXY Cartesian plane. Flying saucer is represented by a circle of radius r, and since modern cities with skyscrapers tend to be regular, every skyscraper is represented with a rectangle whose sides are parallel to OX and OY axes.

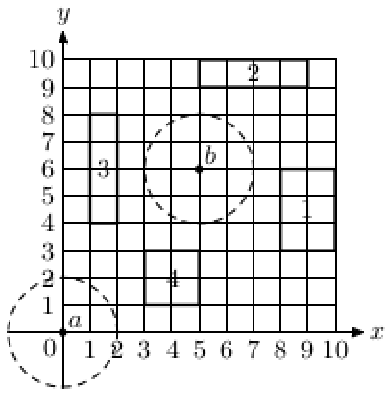
By definition, the location of flying saucer is the location of its center, and the length of the path it travels is the length of the path its center travels. During its mission flying saucer can touch skyscrapers but it cannot intersect them.

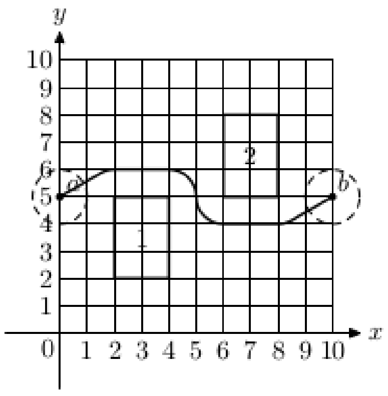
At the first picture a flying saucer of $r = 1$ has to get from point A to point B. The straight dashed line would have been the shortest path if not for skyscraper 1. The shortest way to avoid skyscraper 1 is going around its top right corner, but skyscraper 2 is too close to fly there. Thus, the answer is to go around the bottom left corner of skyscraper 1 for a total path length of 10.570796.

In the second picture it is impossible for a flying saucer of $r = 2$ to get from point A to point B, since all skyscrapers are too close to fly in between them.

In the third picture flying saucer of $r = 1$ has to fly in a slalom-like way around two skyscrapers in order to achieve the shortest path of length 11.652892 between A and B.







Input

The first line of the input file contains integer numbers r and n ($1 \leq r \leq 100$, $0 \leq n \leq 30$), where r is the radius of the flying saucer, and n is the number of skyscrapers. The next line contains four integer numbers x_A, y_A, x_B, and y_B ($-1000 \le x_A, y_A, x_B, y_B \le 1000$), where (x_A, y_A) are the coordinates of the starting point of the flying saucer's mission and (x_B, y_B) are the coordinates of its finishing point.

The following n lines describe skyscrapers. Each skyscraper is represented by four integer numbers x_1, y_1, x_2, and y_2 ($-1000 \le x_1, y_1, x_2, y_2 \le 1000$, $x_1 < x_2$, $y_1 < y_2$) — coordinates of the corners of the corresponding rectangle.

Skyscrapers neither intersect nor touch each other. Starting and finishing points of the flying saucer's mission are valid locations for flying saucer, that is, it does not intersect any skyscraper in those points, but may touch some of them.

Output

Write to the output file text "`no solution`" (without quotes) if the flying saucer cannot reach its finishing point from the starting one. Otherwise, write to the output file a single number — the shortest distance that the flying saucer needs to travel to get from the starting point to the finishing point. Answer has to be precise to at least 6 digits after the decimal point.

Example

standard input	standard output

1 3
2 7 7 1
3 2 6 4
7 5 9 8
1 8 5 9
	10.570796

2 4
0 0 5 6
8 3 10 6
5 9 9 10
1 4 2 8
3 1 5 3
	no solution

1 2
0 5 10 5
2 2 4 5
6 5 8 8
	11.652892

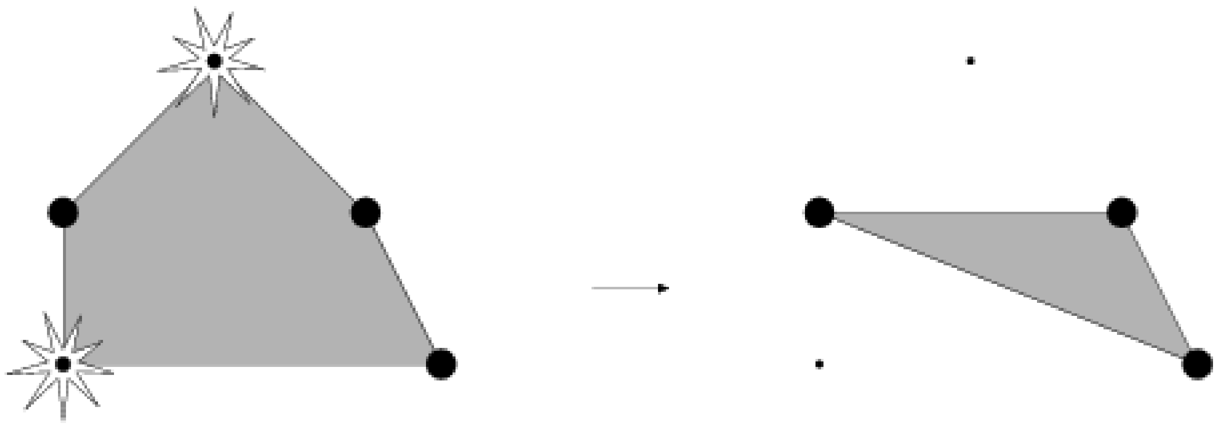
Problem D. Jungle Outpost

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

There is a military base lost deep in the jungle. It is surrounded by n watchtowers with ultrasonic generators. In this problem watchtowers are represented by points on a plane.

Watchtowers generate ultrasonic field and protect all objects that are strictly inside the towers' convex hull. There is no tower strictly inside the convex hull and no three towers are on a straight line.

The enemy can blow up some towers. If this happens, the protected area is reduced to a convex hull of the remaining towers.



The base commander wants to build headquarters inside the protected area. In order to increase its security, he wants to maximize the number of towers that the enemy needs to blow up to make the headquarters unprotected.

Input

The first line of the input file contains a single integer n ($3 \le n \le 50\,000$) — the number of watchtowers. The next n lines of the input file contain the Cartesian coordinates of watchtowers, one pair of coordinates per line. Coordinates are integer and do not exceed 10^6 by absolute value. Towers are listed in the order of traversal of their convex hull in clockwise direction.

Output

Write to the output file the number of watchtowers the enemy has to blow up to compromise headquarters protection if the headquarters are placed optimally.

Example

standard input	standard output

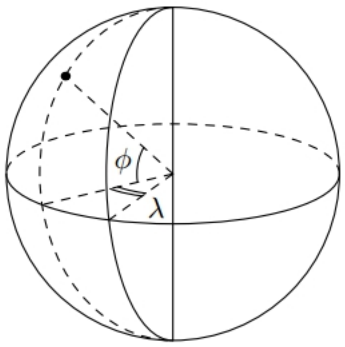
3
0 0
50 50
60 10
	1

5
0 0
0 10
10 20
20 10
25 0
	2

Problem E. Knockdown

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 seconds
Memory limit:	256 mebibytes

The Evil Empire is devising a plan to destroy the world. This plan is called «Operation Knockdown». The plan is to place a number of high-yield nuclear bombs in different places around the world, so that their simultaneous detonation will destroy everything on the planet. For the purpose of planning this operation, bombs have "destruction distance-— the distance from the point of bomb's detonation to all the places where everything is considered destroyed. Places for the bombs have been already selected. Now the Evil Empire wants to minimize the cost of production for the bombs. It is expensive to design atomic bombs tailored for different destruction distances, so the idea is to design a bomb with a specific destruction distance and to produce bombs according to this design for all the selected places. The problem is to find the minimal required destruction distance for the bomb design, so that destruction of the whole world is ensured.



For this problem the world is modeled as a sphere of a unit radius. The coordinates of the selected points for the bombs are specified in geographic coordinate system with latitude $\phi(-90° < \phi < 90°)$ and longitude $\lambda(-180° < \lambda \leqslant 180°)$. Latitude is the angle between a point and the equator, and longitude is the angle between a point and the prime meridian. The bombs are never placed on the poles, so their latitude is always less than 90 degrees by its absolute value. Distances for destruction purposes are measured on the sphere. For example, the distance between the poles is exactly π. The world is considered destroyed if the distance from any point on the sphere to the closest bomb is less or equal to the destruction radius.

Input

The first line of the input file contains a single integer number $n(1 \leqslant n \leqslant 20)$ — the number of the bombs. The following n lines describe the places of the bombs. Each line contains two integer numbers $\phi_i and \lambda_i(-90° < \phi_i < 90°, -180° < \lambda \leqslant 180°)$ — latitude and longitude of the bomb. No two bombs are situated in the same place.

Output

Write to the output file a single number — the minimal destruction radius of the bombs that ensures destruction of the whole world. The answer must be precise up to 10^{-6}.

Example

standard input	standard output

4
59 30
53 83
41 69
41 41
	2.864479

Problem F. Explosion in a Pyramid

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Lich Sandro is standing on the floor of a triangular pyramid at the point $(x_0, y_0, 0)$. The vertices of the pyramid have coordinates $(x_1, y_1, 0)$, $(x_2, y_2, 0)$, $(x_3, y_3, 0)$, (x_4, y_4, H). He wants to fly to the point (x_0, y_0, h) and create a spherical force field centered at that point. A rise to the height z takes z units of mana. Creating a field of radius r takes r units of mana. If the force field has at least one common point with the walls or floor, then the pyramid collapses. Initially Sandro has m units of mana. Is this amount sufficient to break the pyramid with a force sfield?

Input

The first line contains the integers m, h, H ($1 \leqslant m, h, H \leqslant 1\,000$). In the following five lines, you are given the integers x_i, y_i, $0 \leqslant i \leqslant 4$ ($-1\,000 \leqslant x_i, y_i \leqslant 1\,000$). The point (x_0, y_0, h) lies strictly inside the pyramid.

Output

Output "`YES`" if Sandro can break the pyramid with a force field, otherwise output "`NO`".

Example

standard input	standard output

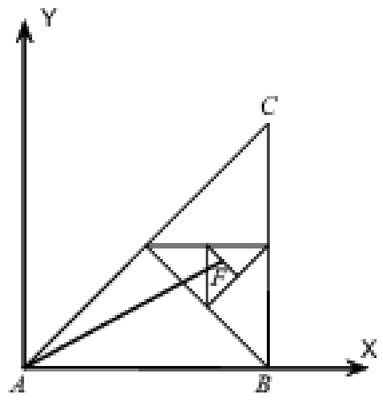
13 6 30
6 6
0 0
0 30
30 0
0 0
	YES

11 6 30
6 6
0 0
0 30
30 0
0 0
	NO

Problem G. Farmland on Mars

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 mebibytes

Alexey Ivanovich decided to sell his farmland on Mars and return to his dear Venus. His farmland has the shape of a flat rectangular triangle ABC, which contains a narrow straight irrigation channel AF. Unfortunately, Alexey Ivanovich didn't manage to find a buyer quickly. It turned out that cottagers wanted to buy only farmlands with through irrigation channel, i.e., a channel having exactly two common points with a border line of a farmland.



Alexey Ivanovich realized that he could divide his farmland into several smaller ones and sell them separately. He decided to divide the initial farmland into two identical triangular farmlands of the same shape by drawing an altitude to the hypotenuse. Then he would consider the half containing F inside and do the same-draw another altitude to the hypotenuse and consider the smaller farmland containing point F. The process would go on until point F is on another altitude, or until the size of the new farmlands is negligibly small.

Help Alexey Ivanovich to count the total area of all resulting farmlands which he will be able to sell.

Input

Let's introduce the coordinates in such a way that point A has the coordinates $(0, 0)$, point B has the coordinates $(10, 0)$, and point C has coordinates $(10, 10)$. The only line contains real numbers x and y that are coordinates of point F $(0 < y < x < 10)$. Numbers x and y are specified with at most three digits after decimal point.

Output

Output the total area of the farmlands fit for sale with absolute or relative error of at most 10^{-3}.

Example

standard input	standard output
`8.125 4.375`	`29.6875`

Problem H. Job or joy

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

How much time can you solve problems non-stop? Sometimes you want to rest, but now is just not the time. It is necessary to collect your thoughts in a fist and snatch victory from your rivals. This problem is designed to separate the merely good teams from the Champions, those who expect to get into the top ten from those who do not intend to take a place below the first. Since the number of enemies who are going to find the author of this tightly introduction to express what they think about him, grows, let's get down to business.

An undirected graph is given. Some queries follow. A query is a pair of vertices. For each query, find out if there exist at least three edge-disjoint paths from the first vertex to the second.

Input

In first line there are two integers n and k ($1 \leqslant n, k \leqslant 100\,000$) — number of vertices and edges in the graph. The next k lines describe edges by pair of connected vertices a, b ($1 \leqslant a, b \leqslant n$). Edges connect different vertices and there is at most one edge between any two vertices. In the next line, there is number of queries q ($0 \leqslant q \leqslant 100\,000$). Each of the next q lines contains two integers a, b ($a \neq b$, $1 \leqslant a, b \leqslant n$) — pair of vertices for which it is required to find out if there exist three distinct paths between them such that none of them share an edge.

Output

For each query, output a single line with either "`Yes`" if such three paths exist, or "`No`" otherwise.

Example

standard input	standard output

6 9
1 2
1 5
1 4
1 6
2 3
3 4
3 5
4 5
4 6
9
1 2
1 3
1 5
2 4
5 6
3 6
3 4
2 6
2 3
	No
	Yes
	Yes
	No
	No
	No
	Yes
	No
	No

Problem I. Area of half-planes' intersection

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

Given n half-planes

$$a_i x + b_i y < c_i,\ 1 \le i \le n.$$

Determine the area of their intersection.

Input

The first line contains the integer n ($1 \le n \le 5 \cdot 10^5$). Each of the next n lines contains three integers a_i, b_i и c_i ($-10^4 \le a_i, b_i, c_i \le 10^4, a_i^2 + b_i^2 > 0$).

Output

Output the area of the intersection of the half-planes or "-1", if it is infinite. Absolute or relative error should not exceed 10^{-6}.

Example

standard input	standard output

3
0 -1 0
-1 0 0
1 1 2
	2.0000000000

1 1 1 0	-1

2
1 2 3
-1 -2 -3
	0

Problem J. Convex Hull 3D

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Given n random points in 3D, find their convex hull.

Input

The first line contains the integer n ($10 \le n \le 10^5$). The next n lines contain the coordinates of the points in the format "x_i y_i z_i". The coordinates are integer, and their absolute values do not exceed 10^4.

It guaranteed, that the points are generated with a random generator. The coordinates are independently and uniformly distributed on the segment $[-10^4, 10^4]$.

Output

Output a single integer, equal to the volume of the convex hull, multiplied by 6.

Example

standard input	standard output

10
1824 1872 -5894
-5449 7960 -8062
8913 -6654 -9250
-2411 4967 -1802
2552 427 7151
6321 -9597 -2138
6728 2094 -1638
2040 542 8862
4309 -7677 -9093
9938 71 -768
	4505172836672

Problem K. Wall

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

A number of points are placed in a four-dimensional space. We can add new points.

At all times all points are surrounded by a single connected wall of a finite size. The wall splits the space into two parts: everything that's within it and all that's outside. The wall is built so that:

1. All the points are within the wall.
2. There is a straight path between any two points within the wall, not crossing the wall.
3. The set of points within the wall is minimal as long as the conditions 1 and 2 are met.

When a new point is added, the wall has to be rebuilt if the point lies outside it. Your task is to define whether wall reconfiguration is necessary for every newly added point.

It is guaranteed that a new point is always built either outside of the existing wall or within it. Moreover, the distance from the new point to the wall is always greater than 10^{-3}. No two points coincide.

Input

The process begins with five points with the coordinates (x_1, y_1, z_1, w_1), (x_2, y_2, z_2, w_2), $\ldots$, (x_5, y_5, z_5, w_5), which are defined in the first five lines of the input file. The initial 4D volume within the wall is strictly positive.

The second line contains an integer N — the number of newly added points ($1 \le N \le 800$). Each of the following N lines contains four integers — the coordinates of a added point. The absolute value of each coordinate does not exceed 5000.

Output

The output file must contain N lines. The K-th line must contain the word "`Rebuild`", if wall reconfiguration is necessary after adding K-th point, and the word "`Ignore`" otherwise ($1 \le K \le N$).

Example

standard input	standard output

0 0 0 0
8 0 0 0
0 8 0 0
0 0 8 0
0 0 0 8
5
1 2 2 2
2 2 3 2
8 8 8 8
3 5 3 4
-1 3 7 2
	Ignore
	Rebuild
	Rebuild
	Ignore
	Rebuild

Problem L. Internal or Exernal?

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

Several convex polygons are put one inside another, so second polygon is put into first one, third — into second one etc. Any next polygon lies strictly inside previous one (i.e. does not intersect or ever touch). You given vertices of all polygons. Your goal is to reconstruct whole picture, i.e. for each vertice tell number of polygon, to which it belongs. Note that a segment and a point are counted as convex polygons with 2 and 1 vertices, respectively.

Input

First line of the input file contains one integer N ($1 \le N \le 2 \cdot 10^4$) — number of vertices. i'th of next N lines contain two integers x_i and y_i — coordinates of i-th point. All coordinates does not exceed 10^4 by absolute value.

Output

Print N lines, one for the each vertice. Each line must contain one integer — number of convex polygon for this vertice (polygons are numbered from 1 starting from biggest one).

Example

standard input	standard output

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