Problem B
Mission Impossible

Source file name: `mission.c`, `mission.cpp`, `mission.java` *or* `mission.pas`

You have been hired to explore enemy territory. It is risky business, you know that. So, you'd better be prepared! The enemy has placed a number of security points all over his country, from which radars are detecting any moving vehicle within their range of cover. Any such detected object will be immediately destroyed. Fortunately enough, you have been given by your government a map of the enemy territory, consisting of coordinates and radius of coverage of each radar. You have also a list of local informers (together with their locations) that you should contact in order to obtain valuable information. Your mission is to try to contact one of these informers, preferably the one with highest *insider-coefficient*. The insider-coefficient of each informer is simply the distance from the informer to the border of the country, where such a distance is defined as the minimum over all distances from the location of the informer to each point of the border. In intuitive sense, the informer with highest insider-coefficient is that who is located as inside the country as possible, and will presumably have more valuable information about the country.

Your first thought is then to design a computer program which will check if there is a path from your initial location, always the point $(2000, 2000)$, to any of the informers' location, without crossing any region which is covered by radar. Whenever possible, the program should indicate which reachable informer is the one to be contacted, according to the insider-coefficient criteria described above.

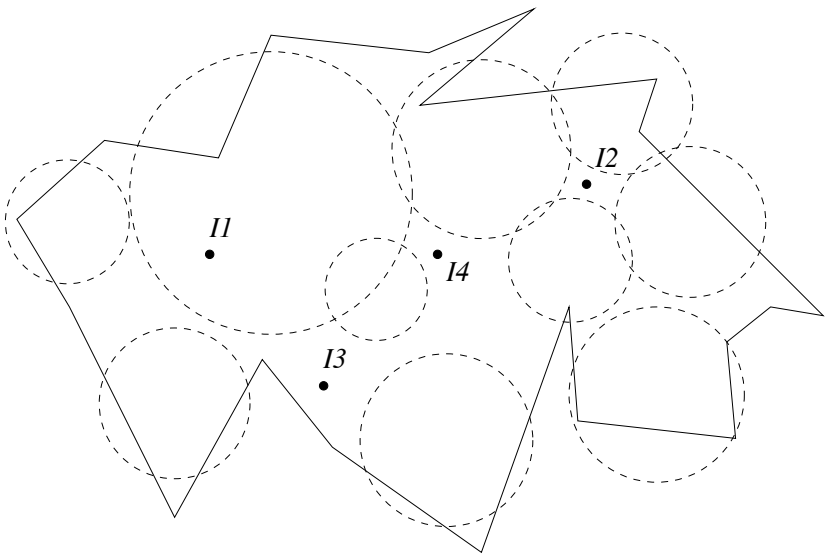


Figure 1: Possible Scenario

The enemy country has the shape of a *simple polygon* (*not necessarily convex*). Recall that a polygon is called simple if it is described by a single, non-intersecting boundary. The borders of the country will be given as a sequence of X,Y-coordinates corresponding to the sequence of vertexes of the polygon. You may assume that all the radar's centres and the informers' coordinates are located within the country's border. Notice, however, that the area covered by the radars might include regions outside the border.

In the sample scenario of Figure 1, informer $I1$ cannot be contacted since he is inside the region covered by radars. The informer $I2$, although outside the radar's region, can't be contacted either since any trip to his location would go through the deadly radar-covered regions. Both informers $I3$ and $I4$ could be contacted, so that informer $I4$ is chosen since his insider-coefficient is greater than that of $I3$.

Input

The input consists of several test cases. The first line of each test case describes the border of the enemy country, in the format

```
B X1 Y1 X2 Y2 ... XB YB
```

where $3 \leq B \leq 1000$ is the number of border points, and each Xi Yi is the coordinate of the i-th point in the border. The border of the country consists of line segments between points i and $i+1$, and between points B and 1. The second line gives the number of informers and their respective positions, in the format

```
N X1 Y1 X2 Y2 ... XN YN
```

where $1 \leq N \leq 1000$ is the number of informers, and Xi Yi is the coordinate of the i-th informer. The third line describes the position and radius of the radars, in the format

```
M X1 Y1 R1 X2 Y2 R2 ... XM YM RM
```

where $1 \leq M \leq 30$ is the number of radars, Xi Yi is the coordinate of the i-th radar, and Ri is the radius of the i-th radar. All the coordinates are integers $0 \leq X, Y \leq 1000$. The radius of the radars are integers in the range $1 \leq R \leq 1000$. A test case where $B = N = M = 0$ indicates the end of the input. This test case must not be processed.

The input must be read from standard input.

Output

For each test case in the input, your program must produce one line containing either "Mission impossible" or "Contact informer K", where "K" is the index of the informer (as given in the input) with highest insider-coefficient which can be reached by the spy without going inside any radar coverage area. If there are more than one informer satisfying this condition, choose the one among them with lowest index.

The output must be written to standard output.

Sample input	**Output for the sample input**

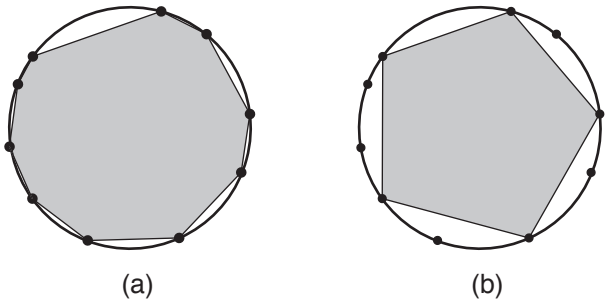
`4 0 0 0 200 200 200 200 0`
`2 70 70 120 120`
`1 100 100 100`
`4 0 0 0 200 200 200 200 0`
`3 100 102 70 80 20 10`
`4 70 70 35 130 70 35 130 130 35 70 130 35`
`0`
`0`
`0`
	`Mission impossible`
	`Contact informer 3`

Problem K
Shrinking Polygons

Source file name: `polygons.c`, `polygons.cpp` *or* `polygons.java`

A polygon is said to be *inscribed* in a circle when all its vertices lie on that circle. In this problem you will be given a polygon inscribed in a circle, and you must determine the minimum number of vertices that should be removed to transform the given polygon into a *regular polygon*, i.e., a polygon that is equiangular (all angles are congruent) and equilateral (all edges have the same length).

When you remove a vertex v from a polygon you first remove the vertex and the edges connecting it to its adjacent vertices w_1 and w_2, and then create a new edge connecting w_1 and w_2. Figure (a) below illustrates a polygon inscribed in a circle, with ten vertices, and figure (b) shows a pentagon (regular polygon with five edges) formed by removing five vertices from the polygon in (a).



In this problem, we consider that any polygon must have at least three edges.

Input

The input contains several test cases. The first line of a test case contains one integer N indicating the number of vertices of the inscribed polygon ($3 \leq N \leq 10^4$). The second line contains N integers X_i separated by single spaces ($1 \leq X_i \leq 10^3$, for $0 \leq i \leq N-1$). Each X_i represents the length of the arc defined in the inscribing circle, clockwise, by vertex i and vertex $(i+1)$ mod N. Remember that an *arc* is a segment of the circumference of a circle; do not mistake it for a *chord*, which is a line segment whose endpoints both lie on a circle.

The end of input is indicated by a line containing only one zero.

The input must be read from standard input.

Output

For each test case in the input, your program must print a single line, containing the minimum number of vertices that must be removed from the given polygon to form a regular polygon. If it is not possible to form a regular polygon, the line must contain only the value `-1`.

The output must be written to standard output.

Sample input	**Output for the sample input**
3 1000 1000 1000 6 1 2 3 1 2 3 3 1 1 2 10 10 40 20 30 30 10 10 50 24 26 0	0 2 -1 5

Problem I
Isosceles Triangles

File code name: `isosceles`

A given triangle can be either equilateral (three sides of the same length), scalene (three sides of different lengths), or isosceles (two sides of the same length and a third side of a different length). It is a known fact that points with all integer coordinates cannot be the vertices of an equilateral triangle.

You are given a set of different points with integer coordinates on the XY plane, such that no three points in the set lay on the same line. Your job is to calculate how many of the possible choices of three points are the vertices of an isosceles triangle.

Input

There are several test cases. Each test case is given in several lines. The first line of each test case contains an integer N indicating the number of points in the set ($3 \leq N \leq 1000$). Each of the next N lines describes a different point of the set using two integers X and Y separated by a single space ($1 \leq X, Y \leq 10^6$); these values represent the coordinates of the point on the XY plane. You may assume that within each test case no two points have the same location and no three points are collinear.

The last test case is followed by a line containing a single zero.

Output

For each test case output a single line with a single integer indicating the number of subsets of three points that are the vertices of an isosceles triangle.

Sample input	**Output for the sample input**
5 1 2 2 1 2 2 1 1 1000 1000000 6 1000 1000 996 1003 996 997 1003 996 1003 1004 992 1000 0	4 10

Problem E
Onion Layers

Source file name: `onion.c`, `onion.cpp`, `onion.java` *or* `onion.pas`

Dr. Kabal, a well recognized biologist, has recently discovered a liquid that is capable of curing the most advanced diseases. The liquid is extracted from a very rare onion that can be found in a country called Onionland. But not all onions of Onionland are worth to take to the lab for processing. Only those onions with an odd number of layers contain the miraculous liquid. Quite an odd discovery!

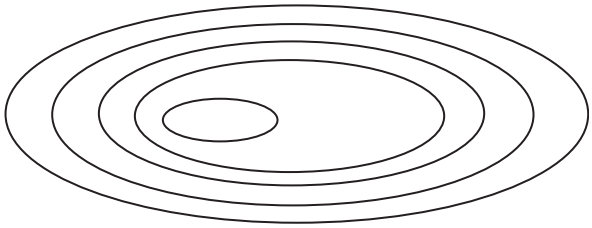


Figure 1: Onion from Onionland

Dr. Kabal has hired a lot of research assistants to collect and analyse onions for him. Since he does not want to share his discovery with the world yet, he didn't tell the assistants to look for onions with an odd number of layers. Instead, each assistant was given the task of collecting onions, and selecting points from each of the layer's outer borders, so that an approximation of the layer structure of the onion can be reconstructed later. Dr. Kabal told the assistants that the next step will be a "complicated analysis" of these points. In fact, all he will do is simply to use the points to count the number of layers in each of the onions, and select the ones with an odd number of layers.

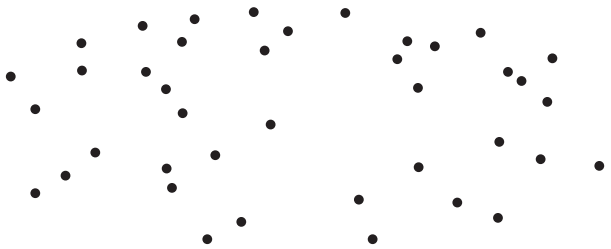


Figure 2: Points collected by an assistant

It is clear that the approximation obtained by Dr. Kabal, from the points collected, might have a different *shape* than the original onion. For instance, only some of the points of the onion shown in Figure 1 would be extracted in the process, giving rise to a set of points as shown in Figure 2. With these points Dr. Kabal will try to approximate the original layers of the onion, obtaining something like what is shown in Figure 3. The approximation procedure followed by Dr. Kabal (whose result is shown in Figure 3) is simply to recursively find nested convex polygons such that at the end every point belongs to precisely one of the polygons. The assistants have been told to select points in such a way that the *number of layers in the*

approximation, if done in this recursive manner, will be the same as in the original onion, so that is fine with Dr. Kabal. The assistants are also aware that they need at least three points to approximate a layer, even the innermost one.

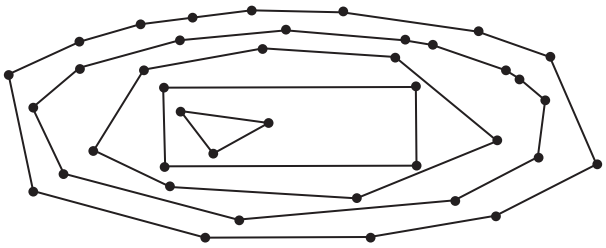


Figure 3: Dr. Kabal's approximation

Your task is to write a program that, given a set of points collected by an assistant (as shown in Figure 2), determines if the respective onion should be taken to the laboratory.

Input

The input contains several test cases. Each test case consists of an integer $3 \leq N \leq 2000$ in a single line, indicating the number of points collected by the assistants. Following, there are N lines, each containing two integers $-2000 \leq X, Y \leq 2000$ corresponding to the coordinates of each point. The input is finished by a problem with $N = 0$ points, which should not be processed.

The input must be read from standard input.

Output

There should be one line of output for each test case in the input. For each test case print the string

```
Take this onion to the lab!
```

if the onion should be taken to the laboratory or

```
Do not take this onion to the lab!
```

if the onion should not be taken to the laboratory.

The output must be written to standard output.

Sample input	**Output for the sample input**
7 0 0 0 8 1 6 3 1 6 6 8 0 8 8 11 2 6 3 2 6 6 0 0 0 11 1 1 1 9 7 1 7 9 8 10 8 0 0	Do not take this onion to the lab! Take this onion to the lab!