109 SCUD Busters

Some problems are difficult to solve but have a simplification that is easy to solve. Rather than deal with the difficulties of constructing a model of the Earth (a somewhat oblate spheroid), consider a pre-Columbian flat world that is a 500 kilometer $\times$ 500 kilometer square.

In the model used in this problem, the flat world consists of several warring kingdoms. Though warlike, the people of the world are strict isolationists; each kingdom is surrounded by a high (but thin) wall designed to both protect the kingdom and to isolate it. To avoid fights for power, each kingdom has its own electric power plant.

When the urge to fight becomes too great, the people of a kingdom often launch missiles at other kingdoms. Each SCUD missile (Sanitary Cleansing Universal Destroyer) that lands within the walls of a kingdom destroys that kingdom's power plant (without loss of life).

Given coordinate locations of several kingdoms (by specifying the locations of houses and the location of the power plant in a kingdom) and missile landings you are to write a program that determines the total area of all kingdoms that are without power after an exchange of missile fire.

In the simple world of this problem kingdoms do not overlap. Furthermore, the walls surrounding each kingdom are considered to be of zero thickness. The wall surrounding a kingdom is the minimal-perimeter wall that completely surrounds all the houses and the power station that comprise a kingdom; the area of a kingdom is the area enclosed by the minimal-perimeter thin wall.

There is exactly one power station per kingdom.

There may be empty space between kingdoms.

Input

The input is a sequence of kingdom specifications followed by a sequence of missile landing locations.

A kingdom is specified by a number N ($3 \leq N \leq 100$) on a single line which indicates the number of sites in this kingdom. The next line contains the x and y coordinates of the power station, followed by $N-1$ lines of x, y pairs indicating the locations of homes served by this power station. A value of -1 for N indicates that there are no more kingdoms. There will be at least one kingdom in the data set.

Following the last kingdom specification will be the coordinates of one or more missile attacks, indicating the location of a missile landing. Each missile location is on a line by itself. You are to process missile attacks until you reach the end of the file.

Locations are specified in kilometers using coordinates on a 500 km by 500 km grid. All coordinates will be integers between 0 and 500 inclusive. Coordinates are specified as a pair of integers separated by white-space on a single line. The input file will consist of up to 20 kingdoms, followed by any number of missile attacks.

Output

The output consists of a single number representing the total area of all kingdoms without electricity after all missile attacks have been processed. The number should be printed with (and correct to) two decimal places.

A Hint: You may or may not find the following formula useful.

Given a polygon described by the vertices $v_0, v_1, \ldots, v_n$ such that $v_0 = v_n$, the signed area of the polygon is given by

$$a = \frac{1}{2}\sum_{i=1}^{n}(x_{i-1}y_i) - (x_i y_{i-1})$$

where the x, y coordinates of $v_i = (x_i, y_i)$; the edges of the polygon are from v_i to v_{i+1} for $i = 0 \ldots n-1$.

If the points describing the polygon are given in a counterclockwise direction, the value of a will be positive, and if the points of the polygon are listed in a clockwise direction, the value of a will be negative.

Sample Input

```
12
3 3
4 6
4 11
4 8
10 6
5 7
6 6
6 3
7 9
10 4
10 9
1 7
5
20 20
20 40
40 20
40 40
30 30
3
10 10
21 10
21 13
-1
5 5
20 12
```

Sample Output

```
70.50
```

477 Points in Figures: Rectangles and Circles

Given a list of figures (rectangles and circles) and a list of points in the x-y plane, determine for each point which figures (if any) contain the point.

Input

There will be $n(\leq 10)$ figures descriptions, one per line. The first character will designate the type of figure ("`r`", "`c`" for rectangle or circle, respectively). This character will be followed by values which describe that figure.

- For a rectangle, there will be four real values designating the x-y coordinates of the upper left and lower right corners.
- For a circle, there will be three real values, designating the x-y coordinates of the center and the radius.

The end of the list will be signalled by a line containing an asterisk in column one.

The remaining lines will contain the x-y coordinates, one per line, of the points to be tested. The end of this list will be indicated by a point with coordinates 9999.9 9999.9; these values should not be included in the output.

Points coinciding with a figure border are not considered inside.

Output

For each point to be tested, write a message of the form:

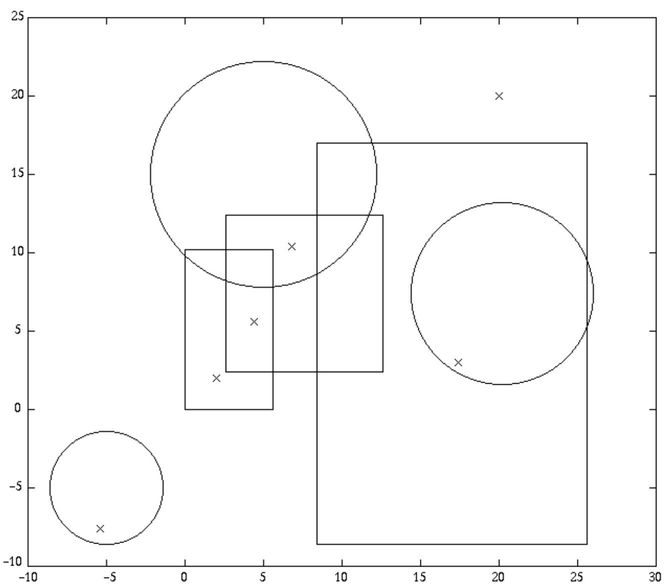
`Point` i `is contained in figure` j

for each figure that contains that point. If the point is not contained in any figure, write a message of the form:

`Point` i `is not contained in any figure`

Points and figures should be numbered in the order in which they appear in the input.

Note: See the picture on the right for a diagram of these figures and data points.



Sample Input

```
r 8.5 17.0 25.5 -8.5
c 20.2 7.3 5.8
r 0.0 10.3 5.5 0.0
c -5.0 -5.0 3.7
r 2.5 12.5 12.5 2.5
c 5.0 15.0 7.2
*
2.0 2.0
4.7 5.3
6.9 11.2
```

```
20.0 20.0
17.6 3.2
-5.2 -7.8
9999.9 9999.9
```

Sample Output

```
Point 1 is contained in figure 3
Point 2 is contained in figure 3
Point 2 is contained in figure 5
Point 3 is contained in figure 5
Point 3 is contained in figure 6
Point 4 is not contained in any figure
Point 5 is contained in figure 1
Point 5 is contained in figure 2
Point 6 is contained in figure 4
```

858 Berry Picking

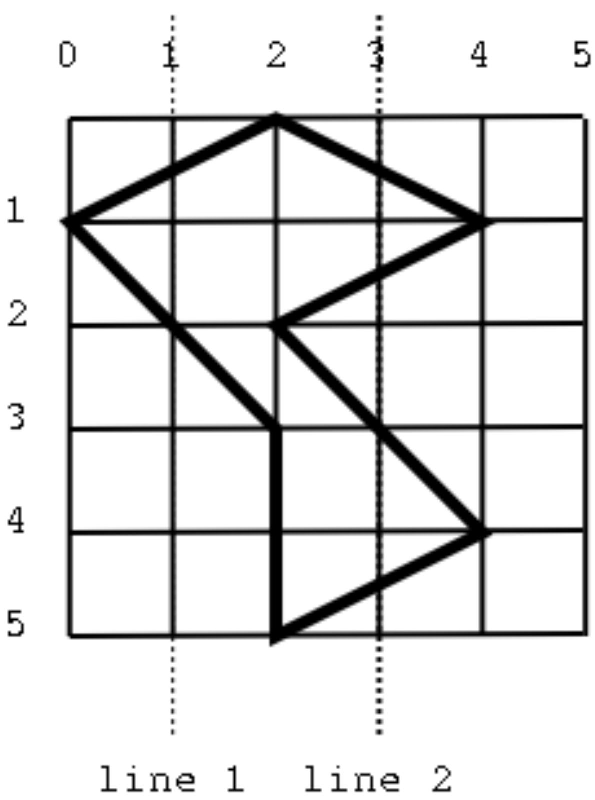
It is not easy for a bird to find food when winter approaches. The fields become empty and all the crops have been harvested. There are only some wild berries that can be picked, but their location is sometimes hidden.

We will help the birds find food. A bird looks for food by flying over a field and scanning it for the edible berries. Suppose that we are given the shape of a berry area in the field and we want to decide whether it is useful for the bird to cross the field along a chosen line. The flight is considered useful if the bird flies over an extension of berries that exceeds some threshold length.

We will view a berry area as a polygon where vertices are approximated by integer values. Birds always follow vertical lines in their flight, and never fly over the vertices of the polygon: in those points there are usually scarecrows that frighten them.

Your task is to decide if the flight along a given line is useful. You will be given a sequence of coordinates defining the berry area, a value for the threshold and the x-coordinate of the line to be followed by the bird.

As an example, the following figure illustrates the location of a berry area in a field, where "line1" and "line2" are possible flight paths. If the threshold is set to 2 length units, line 2 will be an interesting flight path but line 1 will be not.



Given the shape of the berry area, the threshold and the x-coordinate of the line to be followed by the bird, decide whether the line is useful. Assume that the given x-coordinate does not intersect the polygon at any vertex.

Input

The first line of the input contains the number T of test cases, followed by T input blocks.

The first line of each inpt block consists of one integer N, the number of vertices of the polygon representing the berry area. Each of the following N input lines contains the X-Y integer coordinates of one vertex. The following line has an integer indicating the threshold for the decision. The last line of each input block has an integer for the x-coordinate to be evaluated. The set of vertices starts at an arbitrary point in the polygon and there are at most 10000 vertices in a polygon.

Output

For each input block output a single line, containing '`YES`' if the chosen line is useful and '`NO`' otherwise.

Sample Input

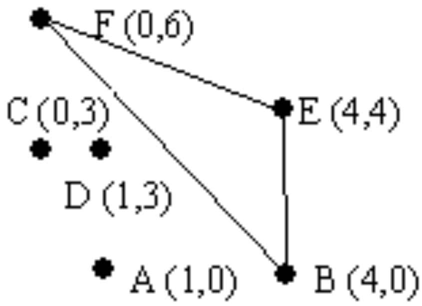
```
2
7
```

```
4 1
2 2
5 3
2 5
2 3
0 1
2 0
2
3
7
4 1
2 2
5 3
2 5
2 3
0 1
2 0
2
3
```

Sample Output

```
YES
YES
```

10112 Myacm Triangles



There has been considerable archeological work on the ancient Myacm culture. Many artifacts have been found in what have been called power fields: a fairly small area, less than 100 meters square where there are from four to fifteen tall monuments with crystals on top. Such an area is mapped out above. Most of the artifacts discovered have come from inside a triangular area between just three of the monuments, now called the power triangle. After considerable analysis archeologists agree how this triangle is selected from all the triangles with three monuments as vertices: it is the triangle with the largest possible area that does not contain any other monuments inside the triangle or on an edge of the triangle. Each field contains only one such triangle.

Archeological teams are continuing to find more power fields. They would like to automate the task of locating the power triangles in power fields. Write a program that takes the positions of the monuments in any number of power fields as input and determines the power triangle for each power field.

A useful formula: the area of a triangle with vertices (x_1, y_1), (x_2, y_2), and (x_3, y_3) is the absolute value of

$$0.5 \times [(y_3 - y_1)(x_2 - x_1) - (y_2 - y_1)(x_3 - x_1)].$$

Input

For each power field there are several lines of data. The first line is the number of monuments: at least 4, and at most 15. For each monument there is a data line that starts with a one character label for the monument and is followed by the coordinates of the monument, which are nonnegative integers less than 100. The first label is A, and the next is B, and so on.

There is at least one such power field described. The end of input is indicated by a 0 for the number of monuments. The first sample data below corresponds to the diagram in the problem.

Output

For each power field there is one line of output. It contains the three labels of the vertices of the power triangle, listed in increasing alphabetical order, with no spaces.

Sample Input

```
6
A 1 0
B 4 0
C 0 3
D 1 3
E 4 4
F 0 6
4
A 0 0
```

```
B 1 0
C 99 0
D 99 99
0
```

Sample Output

```
BEF
BCD
```

10195 The Knights Of The Round Table

King Arthur is planning to build the round table in a new room, but this time he wants a room that have sunlight entering it, so he planned to build a glass roof. He also wishes his round table to shine during the day, specially at noon, so he wants it to be covered totally by the sunlight. But Lancelot wants the glass part of the room roof to be triangular (and nobody knows the reason why, maybe he made a vow or something like that). So, there will be a triangular area in the room which will be all covered by the sunlight at noon and the round table must be build in this area.

Now, King Arthur wants to build the biggest table that he cans such that it fits in the triangular sunlighted area. As he is not very good in geometry, he asked Galahad to help him (Lancelot is very good in geometry, but King Arthur didn't asked Lancelot to help him because he feared that he would come up with another strange suggestion).

Can you help Galahad (since he's not too good with computers) and write a program which gives the radius of the biggest round table that fits in the sunlighted area? You can assume that the round table is a perfect circle.

Input

There'll be an arbitrary number of rooms. Each room is represented by three real numbers (a, b and c), which stand for the sizes of the triangular sunlighted area. No triangle size will be greater than 1000000 and you may assume that $\max(a, b, c) \leq (a + b + c)/2$.

You must read until you reach the end of the file.

Output

For each room configuration read, you must print the following line:

`The radius of the round table is:` r

Where r is the radius of the biggest round table that fits in the sunlighted area, rounded to 3 decimal digits.

Sample Input

```
12.0 12.0 8.0
```

Sample Output

```
The radius of the round table is: 2.828
```

11473 Campus Roads

NGU has a beautiful campus. The authority of NGU has decided to take some more steps about beautification of this campus. One of these steps is **tree plantation** in some roads. To complete this project they will select some roads, and for any road they will also select a number that how many tree will be planted on that road. To make this campus more beautiful they decided in a single road, the gap between any two adjacent tree will be fixed, and this gap will be as large as possible. So If you walk through the road in a constant speed, you will reach a tree after constant intervals.

A road can be described by K points $P_1, P_2, \ldots, P_K$. P_i and P_{i+1} are connected by a straight line. So, it is clear that a road consists of some contiguous straight lines.

In this problem you have to find all the points, where to plant a tree. For the sake of simplicity, we will consider that roads will not have any width.

Input

The first line of input is an integer N ($N < 100$) that indicates the number of roads under this beautification project. From next line N roads is described one by one. Description of each road starts with a line containing two integers K ($2 \leq K \leq 100$) and T ($2 \leq T \leq 100$). Here K represents number of points that represents the road, and T represents the number of trees to be planted. This line is followed by K lines having 2 integers x, y ($-1000.00 \leq x \leq 1000.00$ and $-1000.00 \leq y \leq 1000.00$) each, i-th line of these K lines is the co-ordinate of P_i.

Output

You have to give your output for every roads one by one. Output of each road starts with a line '`Road #`n`:`' (without quotes), here n is the road number. Next T lines will give the co-ordinates of trees to be planted. x, y co-ordinate will be separated by a single space and having two decimal points. You have to represent the points sequentially in the order in which order you will find those trees when you walk through the road P_1 to P_K.

Print a blank line after describing each road.

Hint: You have to maximize the distance between two adjacent trees in a road. So it is sure that first and last tree will be on the first and last point, i.e. in P_1 and P_k.

Sample Input

```
1
5 6
10.00 10.00
20.00 20.00
30.00 10.00
10.00 0.00
9.00 9.00
```

Sample Output

```
Road #1:
10.00 10.00
18.44 18.44
```

```
26.89 13.11
23.26 6.63
12.58 1.29
9.00 9.00
```