Convex Hull Finding

Given a single connected contour, which is either convex or non-convex (concave), use any algorithm to find its **Convex Hull**, i.e., the smallest convex contour enclosing the given shape. If the given contour is convex, then its convex hull is the original contour itself. The maximal size of the shape is 512 $\times$ 512, and the maximal number of the vertices of the shape is 512. Write a program to read the input data (the given shapes) from a disk file, implement your convex hull finding algorithm, and then output the shape data of the results to the standard output.

Input

The order of the vertices is counterclockwise in *X*-*Y* Cartesian Plane (if you consider the origin of the display window is on the upper-left corner, then the orientation of the vertices is clockwise), and none of the neighboring vertices are co-linear. Since all the shapes are closed contours, therefore, the last vertex should be identical to the first vertex. There are several sets of data within a given data file. The negative number `-1` is used to separate the data set.

Line	**Data in**	
Number	**the File**	**Explanation**
1	K	a positive integer showing how many sets of data in this file
2	N	a positive integer showing the number of vertices for the shape
3	X_1 Y_1	two positive integers for the first vertex (X_1, Y_1)
4	X_2 Y_2	two positive integers for the next neighboring vertex (X_2, Y_2)
...		
N+2	X_N Y_N	two positive integers for the last vertex (X_N, Y_N)
N+3	`-1`	Delimiter

N+4	M	a positive integer showing the number of vertices for the next shape
N+5	XX_1 YY_1	two positive integers for the first vertex
...		

Note: Please note that the **Line Number**, **Data in the File** and **Explanation** are not given in the file. They are shown here only to assist you in reading the data.

Output

Output the convex hull of all K input shapes to the standard output. The data format should be the same as the input file. In addition, the vertex with the smallest Y value should be the first point and if there are points with the same Y value, then the smallest X value within those points should be the first point.

Sample Input

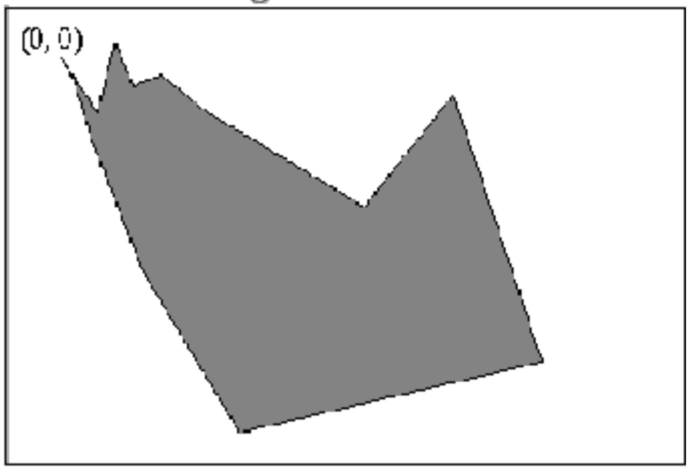
```
3
15
30 30
50 60
60 20
70 45
86 39
112 60
200 113
250 50
300 200
130 240
76 150
47 76
36 40
33 35
30 30
-1
12
50 60
60 20
70 45
100 70
125 90
200 113
250 140
180 170
105 140
79 140
60 85
```

```
50 60
-1
6
60 20
250 140
180 170
79 140
50 60
60 20
```

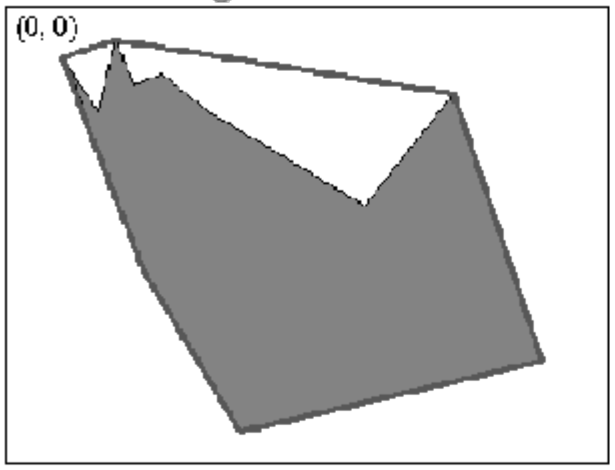
Sample Output

```
3
8
60 20
250 50
300 200
130 240
76 150
47 76
30 30
60 20
-1
6
60 20
250 140
180 170
79 140
50 60
60 20
-1
6
60 20
250 140
180 170
79 140
50 60
60 20
```

The contour shape of the first data
set is shown in figure as follows:

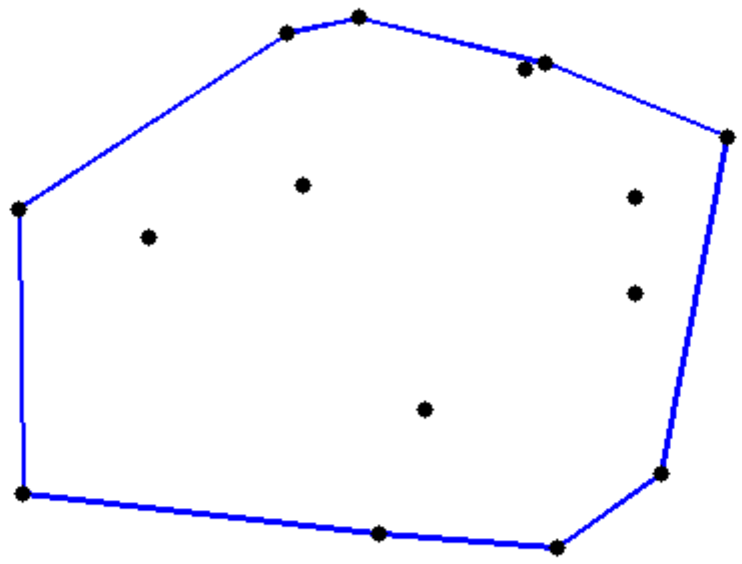


The convex hull of the above shape is shown in the following figure:



Miguel Revilla
2000-08-14

Problem D: Convex Hull



Finding the convex hull of a set of points is an important problem that is often part of a larger problem. There are many algorithms for finding the convex hull. Since problems involving the convex hull sometimes appear in the ACM World Finals, it is a good idea for contestants to know some of these algorithms.

Finding the convex hull of a set of points in the plane can be divided into two sub-tasks. First, given a set of points, find a subset of those points that, when joined with line segments, form a convex polygon that encloses all of the original points. Second, output the points of the convex hull in order, walking counter-clockwise around the polygon. In this problem, the first sub-task has already been done for you, and your program should complete the second sub-task. That is, given the points that are known to lie on the convex hull, output them in order walking counter-clockwise around the hull.

Input Specification

The first line of input contains a single integer, the number of test cases to follow. The first line of each test case contains a single integer 3 <= *n* <= 100000, the number of points. The following *n* lines of the test case each describe a point. Each of these lines contains two integers and either a `Y` or an `N`, separated by spaces. The two integers specify the x- and y-coordinates of the point. A `Y` indicates that the point is on the convex hull of all the points, and a `N` indicates that it is not. The x- and y-coordinates of each point will be no less than -1000000000 and no greater than 1000000000. No point will appear more than once in the same test case. The points in a test case will never all lie on a line.

Sample Input

```
1
5
1 1 Y
1 -1 Y
0 0 N
-1 -1 Y
-1 1 Y
```

Output Specification

For each test case, generate the following output. First, output a line containing a single integer *m*, the number of points on the convex hull. Next output *m* lines, each describing a point on the convex hull, in counter-clockwise order around the hull. Each of these lines should contain the x-coordinate of the point, followed by a space, followed by the y-coordinate of the point. Start with the point on the hull whose x-coordinate is minimal. If there are multiple such points, start with the one whose y-coordinate is minimal.

Output for Sample Input

```
4
-1 -1
1 -1
1 1
-1 1
```

Ondřej Lhoták, Malcolm Sharpe

D - Isolated Segments

Time Limit: 1 sec
Memory Limit: 16MB

You're given n segments in the rectangular coordinate system. The segments are defined by start and end points (X_i,Y_i) and (X_j,Y_j) (1 <= i, j <= n). Coordinates of these points are integer numbers with real value smaller then 1000. Length of each segment is positive.

When 2 segments don't have a common point then it is said that segments don't collide. In any other case segments collide. Be aware that segments collide even if they have only one point in common.

Segment is said to be isolated if it doesn't collide with all the other segments that are given, i.e. segment i is isolated when for each 1 <= j <= n, (i != j), segments i and j don`t collide. You are asked to find number T - how many segments are isolated.

INPUT:

First line of input contains number N (N <= 50), then tests follow. First line of each test case contains number M (M <= 100) - the number of segments for this test case to be considered. For this particular test case M lines follow each containing a description of one segment. Segment is described by 2 points: start point (X_{pi},Y_{pi}) and end point (X_{ei},Y_{ei}). They are given in such order: X_{pi} Y_{pi} X_{ei} Y_{ei}

OUTPUT:

For each test case output one line containing number T.

SAMPLE INPUT:

```
6
3
0 0 2 0
1 -1 1 1
2 2 3 3
2
0 0 1 1
1 0 0 1
2
0 0 0 1
0 2 0 3
2
```

```
0 0 1 0
1 0 2 0
2
0 0 2 2
1 0 1 1
2
1 3 1 5
1 0 1 6
```

SAMPLE OUTPUT:

```
1
0
2
0
0
0
```

Problem setters: Aleksej Viktorchik, Leonid Shishlo.
Huge Easy Contest #1

Problem E: Tunnelling the Earth

There are different methods of transporting people from place to place: cars, bikes, boats, trains, planes, etc. For very long distances, people generally fly in a plane. But this has the disadvantage that the plane must fly around the curved surface of the earth. A distance travelled would be shorter if the traveller followed a straight line from one point to the other through a tunnel through the earth.

For example, travelling from Waterloo to Cairo requires a distance of 9293521 metres following the great circle route around the earth, but only 8491188 metres following the straight line through the earth.

For this problem, assume that the earth is a perfect sphere with radius of 6371009 metres.

Input Specification

The first line of input contains a single integer, the number of test cases to follow. Each test case is one line containing four floating point numbers: the latitude and longitude of the origin of the trip, followed by the latitude and longitude of the destination of the trip. All of these measurements are in degrees. Positive numbers indicate North latitude and East longitude, while negative numbers indicate South latitude and West longitude.

Sample Input

```
1
43.466667 -80.516667 30.058056 31.228889
```

Output Specification

For each test case, output a line containing a single integer, the difference in the distance between the two points following the great circle route around the surface of the earth and following the straight line through the earth, in metres. Round the difference of the distances to the nearest integer number of metres.

Output for Sample Input

```
802333
```

Ondřej Lhoták

Problem D
Trees on My Island

Input: standard input
Output: standard output

I have bought an island where I want to plant trees in rows and columns. So, the trees will form a rectangular grid and each of them can be thought of having integer coordinates by taking a suitable grid point as the origin.

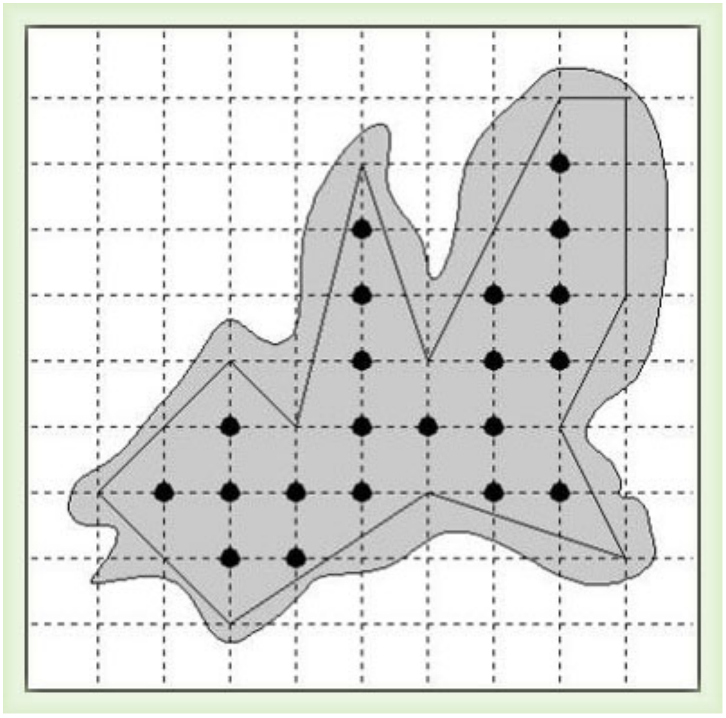


Figure: A sample of my island

But, the problem is that the island itself is not rectangular. So, I have identified a simple polygonal area inside the island with vertices on the grid points and have decided to plant trees on grid points lying strictly inside the polygon.

Now, I seek your help for calculating the number of trees that can be planted on my island.

Input

The input file may contain multiple test cases. Each test case begins with a line containing an integer *N* (3 <= *N* <= 1,000) identifying the number of vertices of the polygon. The next *N* lines contain the vertices of the polygon either in clockwise or in anti-clockwise direction. Each of these *N* lines contains two integers identifying the *x* and *y*-coordinates of a vertex. You may assume that none of the coordinates will be larger than 1,000,000 in absolute values.

A test case containing a zero for *N* in the first line terminates the input.

Output

For each test case in the input print a line containing the number of trees that can be planted inside the polygon.

Sample Input

```
12
3 1
6 3
9 2
8 4
9 6
9 9
8 9
6 5
5 8
4 4
3 5
1 3
12
1000 1000
2000 1000
4000 2000
6000 1000
8000 3000
8000 8000
7000 8000
5000 4000
4000 5000
3000 4000
3000 5000
1000 3000
0
```

Sample Output

```
21
25990001
```

Rezaul Alam Chowdhury

What we see is often what we look for.

Problem M

Rope Crisis in Ropeland !

Input: standard input
Output: standard output
Time Limit: 2 seconds

This is a story of Ropeland where rope pulling is a very popular game (like cricket in Bangladesh). Perhaps you know the game rope pulling: two groups of players hold two ends of a rope. When a certain signal is given they start pulling ropes. The group that can snatch the rope from the other group is declared winner. Today is a very happy day in Ropeland as they have got rope status (something like Bangladesh's test status). So the people of Ropeland are on the street and they are willing to be engaged in rope pulling. But the shops in the city fail to supply enough rope and so now a rope crisis has begun. The King of the country declares a new rule that two groups will not be allowed to buy more ropes than what they require.

The problem is that rope-pulling takes place in a large hall room that has a large round pillar in the middle with certain radius. So if two groups are on the opposite side of the pillar their pulled rope is never in a straight line. Given the position of the two groups you are to find out the minimum length of rope required by them to start rope-pulling. You can assume that a point represents the position of each group.

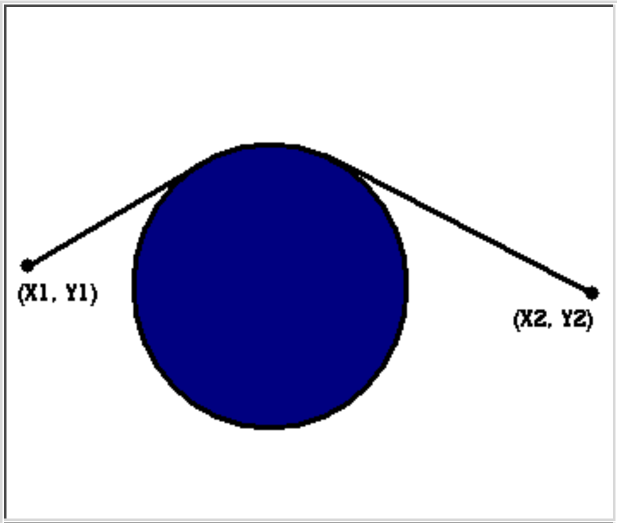
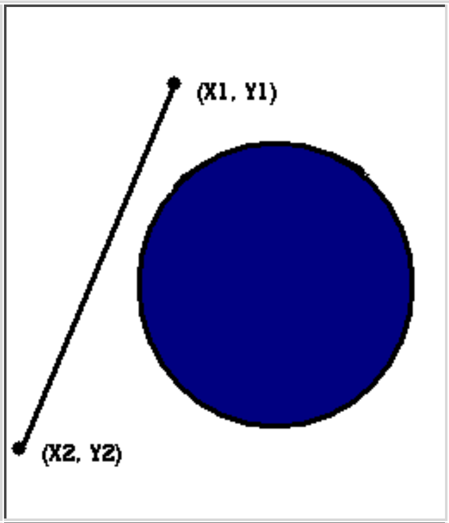
	

Fig 1: Situation when two groups have the round pillar between them. The pulled rope is never a straight line	Fig 2: Situation when the two groups don't have the round pillar between them

Input

The first line of the input file contains an integer **N,** which tells how many sets of input are there. Next there are **N** lines of input.

Each line contains five numbers **X1, Y1, X2, Y2** and **R (> 0)** where **(X1, Y1)** and **(X2, Y2)** are the coordinates of the two groups and **R** is the radius of the pillar. The coordinate of the center of the pillar is always the origin. You can also assume that none of the coordinates will be inside the circle. All input numbers except **N** are floating point numbers and none of their absolute value is greater than **10000**.

Output

For each set of input output a floating-point number in a new line rounded to the third digit after the decimal point and this number denotes the minimum length of the rope required.

Sample Input

```
2
1 1 -1 -1 1
1 1 -1 1 1
```

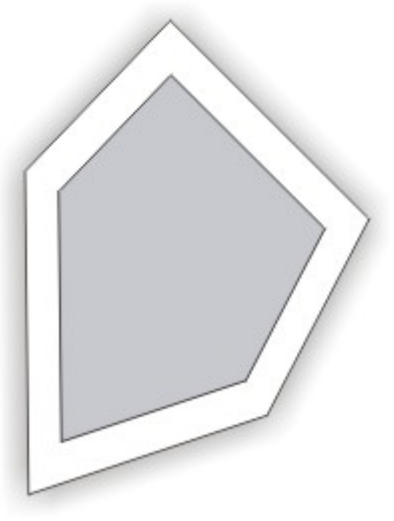
Sample Output

```
3.571
2.000
```

Rezaul Alam Chowdhury (Idea generator and judge solution writer) & **Shahriar Manzoor** (Problem statement writer)

“It may seem weird to many, but there are more triangles
than there are points in this universe.”

Problem D: Cutting tabletops



Bever Lumber hires beavers to cut wood. The company has recently received a shippment of tabletops. Each tabletop is a convex polygon. However, in this hard economic times of cutting costs the company has ordered the tabletops from a not very respectable but cheap supplier. Some of the tabletops have the right shape but they are slightly too big. The beavers have to chomp of a strip of wood of a fixed width from each edge of the tabletop such that they get a tabletop of a similar shape but smaller. Your task is to find the area of the tabletop after beavers are done.

Input consists of a number of cases each presented on a separate line. Each line consists of a sequence of numbers. The first number is **d** the width of the strip of wood to be cut off of each edge of the tabletop in centimeters. The next number **n** is an integer giving the number of vertices of the polygon. The next **n** pairs of numbers present $\mathbf{x}_i$ and $\mathbf{y}_i$ coordinates of polygon vertices for 1 <= i <= **n** given in clockwise order. A line containing only two zeroes terminate the input.

d is much smaller than any of the sides of the polygon. The beavers cut the edges one after another and after each cut the number of vertices of the tabletop is the same.

For each line of input produce one line of output containing one number to three decimal digits in the fraction giving the area of the tabletop after cutting.

Sample input

```
2 4 0 0 0 5 5 5 5 0
1 3 0 0 0 5 5 0
1 3 0 0 3 5.1961524 6 0
3 4 0 -10 -10 0 0 10 10 0
0 0
```

Output for sample input

```
1.000
1.257
```

```
2.785
66.294
```

Problem Setter: Piotr Rudnicki