Algorithmique effective – Partiel 2
Mardi 14 février 2006, 16h00 – 19h00

1 Powerful

We say that x is a perfect square if, for some integer b, $x = b^2$. Similarly, x is a perfect cube if, for some integer b, $x = b^3$. More generally, x is a perfect pth power if, for some integer b, $x = b^p$. Given an integer x you are to determine the largest p such that x is a perfect pth power.

Each test case is given by a line of input containing x. The value of x will have magnitude at least 2 and be within the range of a (32-bit) int in C, C++, and Java. A line containing 0 follows the last test case.

For each test case, output a line giving the largest integer p such that x is a perfect pth power.

Sample Input

```
17
1073741824
25
0
```

Output for Sample Input

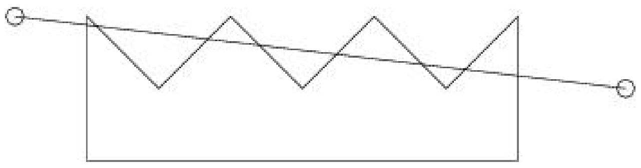
```
1
30
2
```

2 Intersection length

Given is a simple but not necessarily convex polygon. Given is also a line in the plane. If the polygon is cut along the line then we may get several smaller polygons. Your task is to find the length of the cut, that is the total length of the segments in the intersection of the line and the polygon.

Input consists of a number of cases. The data of each case appears on a number of input lines, the first of which contains two non negative integers n and m giving the number of the vertices of the polygon and the number of cutting lines to consider, $3 \leq n \leq 1000$. The following n lines contain coordinates of the vertices of the polygon; each line contains the x and y coordinates of a vertex. The vertices are given either in clockwise or counterclockwise order. Each of the following m lines of input contains four numbers; these are x and y coordinates of the two points defining the cutting line. If a vertex of the polygon is closer than 10e-8 to the cutting line then we consider that the vertex lies on the cutting line.

Input is terminated by a line with n and m equal to 0.



For each cutting line, print the total length of the segments in the intersection of the line and the polygon defined for this test case, with 3 digits after the decimal point. Note: the perimiter of a polygon belongs to the polygon.

The picture above illustrates the first cutting line for the polygon from the sample.

Sample input

```
9 5
0 0
0 2
1 1
2 2
3 1
4 2
5 1
6 2
6 0
-1 2 7.5 1
0 1 6 1
0 1.5 6 1.5
0 2 6 1
0 0 0 2
0 0
```

Output for sample input

```
2.798
6.000
3.000
2.954
2.000
```

3 A polygon in balance

Find out the center of masses of a convex polygon.

Input

A series of convex polygons, defined as a number n ($n \leq 100$) stating the number of points of the polygon, followed by n different pairs of integers (in no particular order), denoting the x and y coordinates of each point. The input is finished by a fake "polygon" with m ($m < 3$) points, which should not be processed. No three points are aligned in any polygon.

Output

For each polygon, a single line with the coordinates x and y of the center of masses of that polygon, rounded to three decimal digits.

Sample Input

```
4 0 1 1 1 0 0 1 0
3 1 2 1 0 0 0
7
-4 -4
-6 -3
-4 -10
-7 -12
-9 -8
-3 -6
-8 -3
1
```

Sample Output

```
0.500 0.500
0.667 0.667
-6.102 -7.089
```

4 Rebuilding a bone

Statement of the problem

We want to reconstruct a surface from a two polygons. Think of it as if we sliced a volume at two z-coordinates, converted these two planar contours into polygons, and we would now like to rebuild a surface of the body between the polygonal cuts from these polygons. In this problem, the resulting surface consists of triangles only.

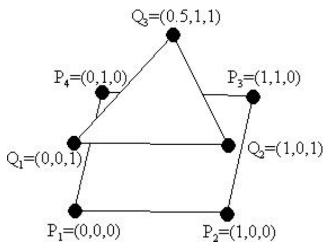


Figure 1

In figure 1, the objective is to "stitch" the square with the triangle. Notice that vertices of the tile triangles which combines these two contour polygons is vertices of the contour polygons, with the vertices of each tile taken two from one polygon and one from the other.

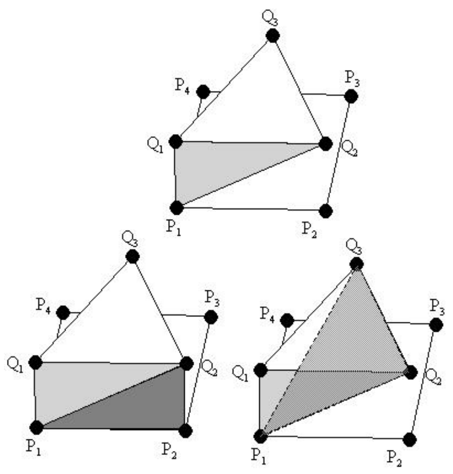


Figure 2

Now suppose we have decided to form a triangle using P_1, Q_1, Q_2, shown in the upper picture of Figure 2. If the vertices of the contour are in counterclockwise order and we are forming triangles in counterclockwise order, we now have two options. Since the surface built must be without "holes", P_1 and Q_2 must be part of the next triangle. This leaves us with either taking the the next P-vertex, shown in the left bottom picture of Figure 2, or taking the next Q-vertex, shown in the right bottom picture.

Since each triangle uses one new vertex from the contour polygons, the resulting surface will consist of 7 $(= 4 + 3)$ triangles.

We would like to know the minimum possible area of such a surface. *One way* of solving this problem is to form a matrix. Suppose the number of vertices of the polygons are m and n. Then build a matrix with m rows and n columns. The position in the matrix corresponds to the two vertices that the next, new, triangle has in common with the last one. A movement to the right means that we are constructing a new triangle using a new vertex from the second polygon (Q), while a movement downwards means a new triangle using a vertex from the first polygon (P). The movement wraps at the rightmost column and the downmost row.

In the example, we started with the points P_1 and Q_1, and used the point Q_2 to form a triangle. Then we could either pick P_2 as our next point, a downward movement, or pick Q_3 as our next point, a rightward movement. Of course, the path must end up at the starting position. For each movement, a cost is associated, namely the area of the triangle.

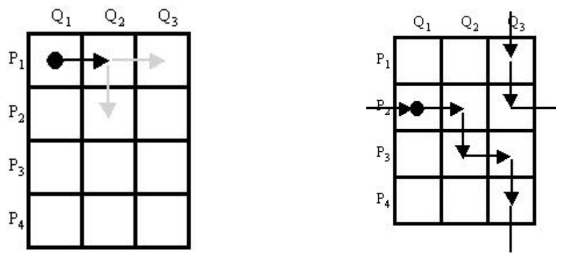


Figure 3

So, the optimization problem can be restated as the problem of finding a closed path with the minimum sum of the costs in the matrix grid, where each horizontal line on the grid is crossed exactly once by the path, and where each vertical line on the grid is crossed exactly once by the path, as in the right of Figure 3.

Hint.

One simple formula to calculate the area of a triangle is Heron's formula: $\sqrt{p(p-a)(p-b)(p-c)}$, where a, b and c are the lengths of the sides, and p is the semiperimeter $(a+b+c)/2$.

Input

The input starts with a single number on a line, N, which stands for the number of test cases. Following this line are the N test cases. Each test case starts with three numbers m, n, z_1. m and n are integers, greater than 2 but less than or equal to 20. m stands for the number of points in the first polygon and n stands for the number of points in the second polygon. z_1 is a real positive number which stands for the z-coordinates for the vertices in the second polygon. The z-coordinates for the vertices in the first polygon is zero. Following this line are $m+n$ lines with x- and y-coordinates of the simple, not necessarily convex, polygons in counterclockwise order. All coordinates are real numbers. First are the coordinates for the first polygon, followed by the coordinates for the second polygon.

Output

For each test case, output the minimum area of the surface connecting the both polygons, using five decimal digits. However, trailing zeros may be omitted. The output is a real number. A relative error of up to 0.01% is allowed.

Example

```
Input:         Output:
2              3.72474
4 3 1          8.00000
0 0
1 0
1 1
0 1
0 0
1 0
0.5 1
4 4 2
0 0
1 0
1 1
0 1
0 0
1 0
1 1
0 1
```

5 Planar tree

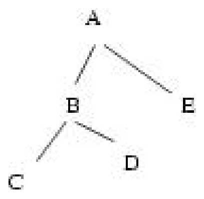
The following grammar describes a textual notation for a tree with *(not necessarily unique)* vertex labels:

```
tree ::= label
tree ::= label ( subtrees )
subtrees ::= tree
subtrees ::= subtrees , tree
label ::= A | B | C | ... | Z
```

That is, the representation of a tree consists of a label (which is an uppercase letter) or a label followed by a bracketed ordered list of trees separated by commas.

In order to draw such a tree on paper, we must write each label on the page, such that the labels for the subtrees of a vertex are positioned counter-clockwise about the vertex. The labels must be positioned such that non-intersecting line segments connect each vertex to each of its subtrees. That is to say, we draw the normal planar representation of the tree, preserving the order of subtrees. Beyond these constraints, the position, shape, and size of the representation is arbitrary.

For example, a possible graphical representation for `A(B(C,D),E)` is



Given the textual representation for two trees, you are to determine whether or not they are equivalent. That is, do they share a common paper representation?

The first line of input contains t, the number of test cases. Each test case consists of two lines, each specifying a tree in the notation described above. Each line will contain at most 200 characters, and no white space. For each test case, output a line containing "same" or "different" as appropriate.

Sample Input

```
2
A(B(C,D),E)
E(A,B(C,D))
A(B(C,D),E)
E(A(B(C,D)))
```

Output for Sample Input

```
different
same
```