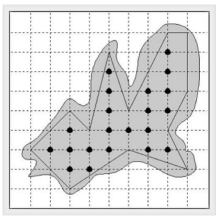
L3/M1, 2005–2006

Algorithmique effective – TD 5
Géométrie, suite

1 Structure arborescente

Exercice 1
Trees on My Island. PC : 111407 ; UVa : 10088.
I have bought an island where I want to plant trees in rows and columns. The trees will be planted to form a rectangular grid, so each can be thought of as having integer coordinates by taking a suitable grid point as the origin.


A sample of my island.

However, my island is not rectangular. 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.

I seek your help in calculating the number of trees that can be planted.

Input

The input file may contain multiple test cases. Each test case begins with a line containing an integer N ($3 \leq N \leq 1000$) identifying the number of vertices of the polygon. The next N lines contain the vertices of the polygon in either the clockwise or counterclockwise direction. Each of these N lines contains two integers identifying the x- and y-coordinates of a vertex. You may assume that the absolute value of all coordinates will be no larger than 1,000,000.

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

Output

For each test case, 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
```

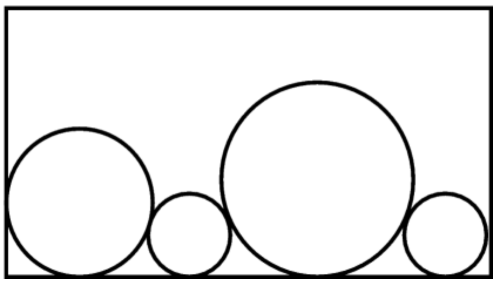
Sample Output

```
21
25990001
```

2 Le camembert commence à sentir...

Exercice 2
How Big is It ? PC : 111308 ; UVa : 10012.
Ian is going to California and has to pack his things, including his collection of circles. Given a set of circles, your program must find the smallest rectangular box they fit in.

All circles must touch the bottom of the box. The figure below shows an acceptable packing for a set of circles, although it may not be the optimal packing for these particular circles. In an ideal packing, each circle should touch at least one other circle, but you probably figured that out.



Input

The first line of input contains a single positive decimal integer n, $n \leq 50$. This indicates the number of test cases to follow. The subsequent n lines each contain a series of numbers separated by spaces. The first number on each of these lines is a positive integer m, $m \leq 8$, which indicates how many other numbers appear on that line. The next m numbers on the line are the radii of the circles which must be packed in a single box. These numbers need not be integers.

Output

For each test case, your program must output the size of the smallest rectangle which can pack the circles. Each case should be output on a separate line by itself, with three places after the decimal point. Do not output leading zeroes unless the number is less than 1, e.g., 0.543.

Sample Input

```
3
3 2.0 1.0 2.0
4 2.0 2.0 2.0 2.0
3 2.0 1.0 4.0
```

Sample Output

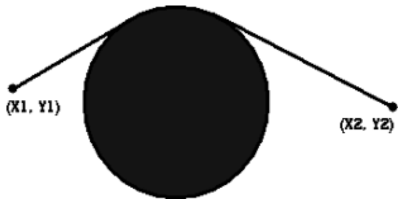
```
9.657
16.000
12.657
```

3 Cordialement

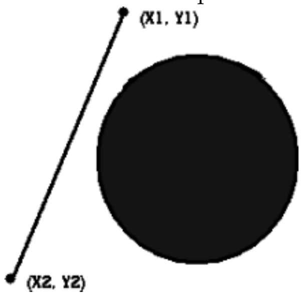
Exercice 3
Rope Crisis in Ropeland ! PC : 111302 ; UVa : 10180
Rope-pulling (also known as tug of war) is a very popular game in Ropeland, just like cricket is in Bangladesh. Two groups of players hold different ends of a rope and pull. The group that snatches the rope from the other group is declared winner.

Due to a rope shortage, the king of the country has declared that groups will not be allowed to buy longer ropes than they require.

Rope-pulling takes place in a large room, which contains a large round pillar of a certain radius. If two groups are on the opposite side of the pillar, their pulled rope cannot be a straight line. Given the position of the two groups, find out the minimum length of rope required to start rope-pulling. You can assume that a point represents the position of each group.



Two groups with the round pillar between them.



Two groups unaffected by the pillar.

Input

The first line of the input file contains an integer N giving the number of input cases. Then follow N lines, each containing five numbers X_1, Y_1, X_2, Y_2, and R, where (X_1, Y_1) and (X_2, Y_2) are the coordinates of the two groups and $R > 0$ is the radius of the pillar.

The center of the pillar is always at the origin, and you may assume that neither team starts in the circle. All input values except for N are floating point numbers, and all have absolute value $\leq$ 10,000.

Output

For each input set, output a floating point number on a new line rounded to the third digit after the decimal point denoting the minimum length of rope required.

Sample Input

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

Sample Output

```
3.571
2.000
```

4 Dessert

En DM pour la prochaine fois (mardi 10 janvier)... bonnes vacances :-)

1. On demande de résoudre le problème des pizzas ci-joint (UVa : 565).
2. Puis on étudiera la transition de phase sur des pizzas aléatoires, courbes à l'appui. Pour cela, on modifiera d'abord le programme précédent afin qu'il accepte plus que 12 personnes (c'est-à-dire plus de *topping constraint lists*), par exemple 150 — mais il n'est plus nécessaire de lire les instances sur l'entrée standard (vous pouvez directement fournir au programme des instances calculées par vos soins).

   Ensuite, on générera des instances aléatoires, de m personnes, chacune fournissant une contrainte de 3 ingrédients, uniformément parmi n ingrédients. Par exemple, pour $m = 5$ et $n = 4$, on pourrait avoir l'entrée suivante :

   ```
   +A-B-C;
   +B+C+D;
   -A-C-D;
   -A+B+D;
   +A+B-C;
   .
   ```

   Chacun aura reconnu qu'il s'agit alors du problème 3-SAT, où m est le nombre de clauses et n le nombre de variables. Soit $\alpha = m/n$. On étudiera la proportion de pizzas aléatoires que l'on peut satisfaire, à α constant, pour n de l'ordre de 15 ou 16. On pourra tracer cette proportion en fonction de α, pour α variant de 0 à 10. Par exemple, incrémenter α de 0,1 en 0,1, fixer n à 15 et m à $\alpha n = 15\alpha$, et étudier la proportion d'instances aléatoires satisfaisables sur un échantillon de 1000 instances aléatoires. On s'intéressera plus particulièrement aux valeurs de α proches de 4,3.

   Enfin, on tracera également le temps de calcul moyen de votre programme sur une instance aléatoire en fonction de α.

   Pour générer des nombres pseudo-aléatoires, utiliser `lrand48()` (utiliser la commande `man lrand48` pour plus de précisions).