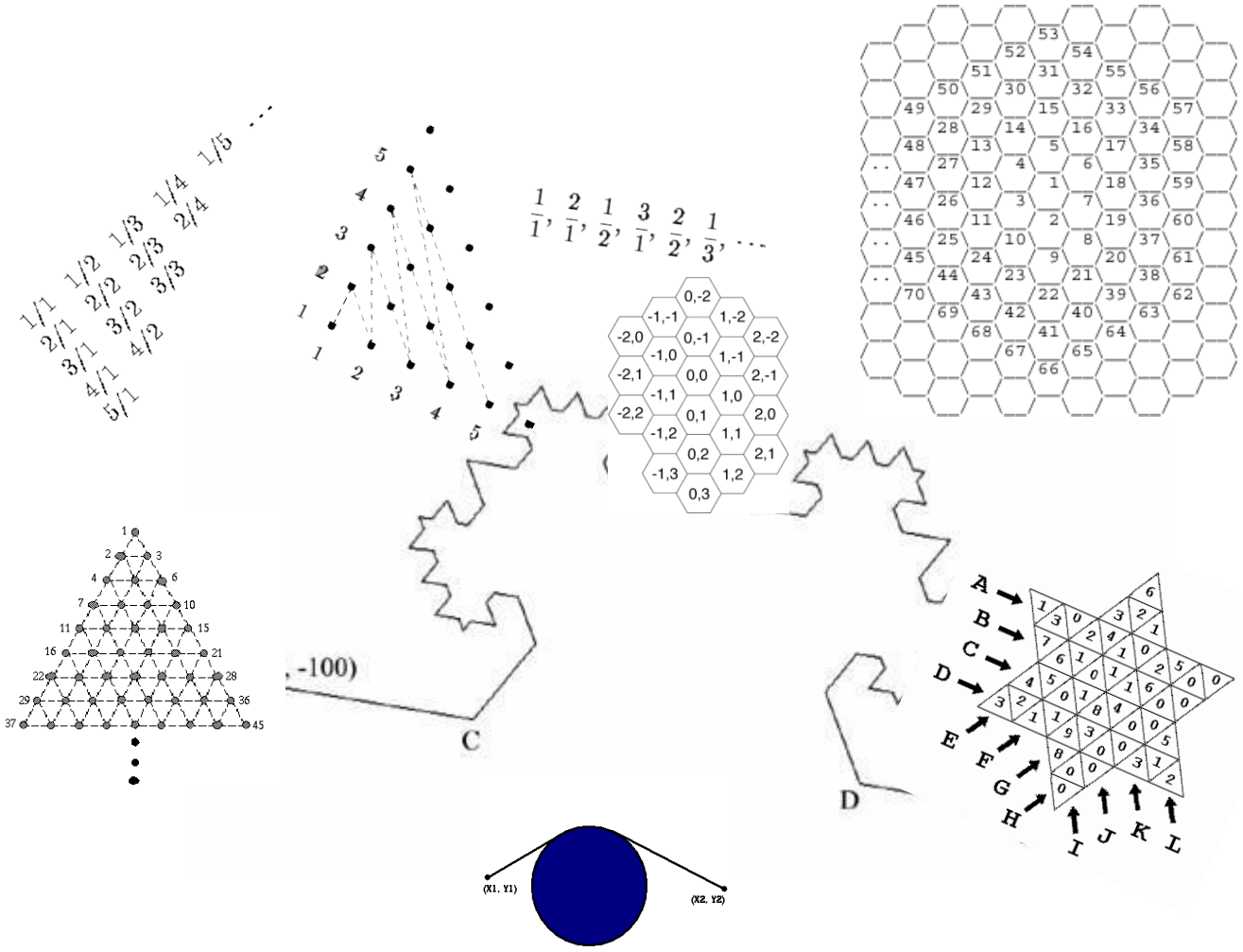
acm International Collegiate Programming Contest

IBM event sponsor

ACM Training

6. Übungsblatt, 2010-10-26



264 - Count on Cantor

Time limit: 3.000 seconds

One of the famous proofs of modern mathematics is Georg Cantor's demonstration that the set of rational numbers is enumerable. The proof works by using an explicit enumeration of rational numbers as shown in the diagram below.

```
1/1  1/2  1/3  1/4  1/5  ...
2/1  2/2  2/3  2/4
3/1  3/2  3/3
4/1  4/2
5/1
```

In the above diagram, the first term is 1/1, the second term is 1/2, the third term is 2/1, the fourth term is 3/1, the fifth term is 2/2, and so on.

Input and Output

You are to write a program that will read a list of numbers in the range from 1 to 10^7 and will print for each number the corresponding term in Cantor's enumeration as given below. No blank line should appear after the last number.

The input list contains a single number per line and will be terminated by end-of-file.

Sample input

```
3
14
7
```

Sample output

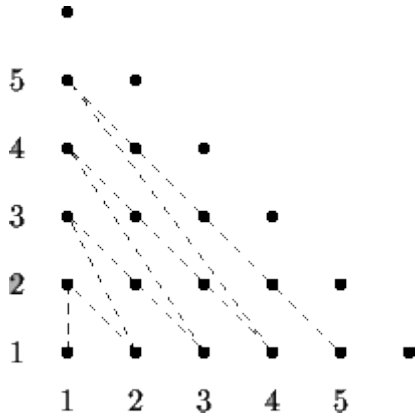
```
TERM 3 IS 2/1
TERM 14 IS 2/4
TERM 7 IS 1/4
```

880 - Cantor Fractions

Time limit: 3.000 seconds

Background

In the late XIXth century the German mathematician George Cantor argued that the set of positive fractions $\mathbf{Q}^+$ is equipotent to the set of positive integers **N**, meaning that they are both infinite, but of the same class. To justify this, he exhibited a mapping from **N** to $\mathbf{Q}^+$ that is onto. This mapping is just *traversal* of the **N** x **N** plane that covers all the pairs:



The first fractions in the Cantor mapping are:

$$\frac{1}{1}, \frac{2}{1}, \frac{1}{2}, \frac{3}{1}, \frac{2}{2}, \frac{1}{3}, \ldots$$

Problem

Write a program that finds the *i*-th Cantor fraction following the mapping outlined above.

Input

The inputs consists of several lines with a positive integer number *i* each one.

Output

The output consists of a line per input case, that contains the *i*-th fraction, with numerator and denominator separed by a slash (/). The fraction should **not** be in the most simple form.

Sample Input
6

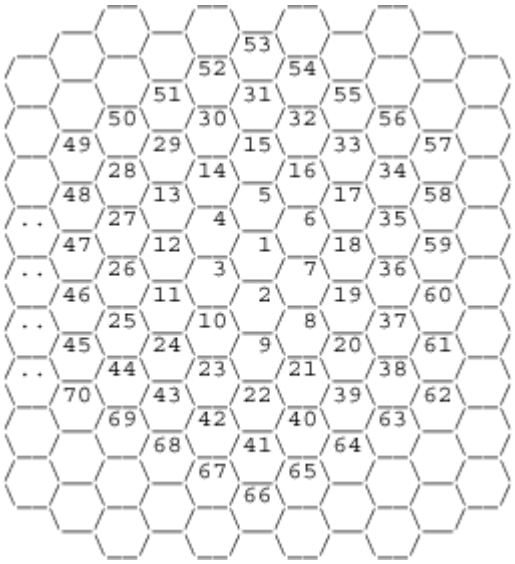
Sample Output
1/3

808 - Bee Breeding

Time limit: 3.000 seconds

Professor B. Heif is conducting experiments with a species of South American bees that he found during an expedition to the Brazilian rain forest. The honey produced by these bees is of superior quality compared to the honey from European and North American honey bees. Unfortunately, the bees do not breed well in captivity. Professor Heif thinks the reason is that the placement of the different maggots (for workers, queens, etc.) within the honeycomb depends on environmental conditions, which are different in his laboratory and the rain forest.

As a first step to validate his theory, Professor Heif wants to quantify the difference in maggot placement. For this he measures the distance between the cells of the comb into which the maggots are placed. To this end, the professor has labeled the cells by marking an arbitrary cell as number 1, and then labeling the remaining cells in a clockwise fashion, as shown in the following figure.



For example, two maggots in cells 19 and 30 are 5 cells apart. One of the shortest paths connecting the two cells is via the cells 19 - 7 - 6 - 5 - 15 - 30, so you must move five times to adjacent cells to get from 19 to 30.

Professor Heif needs your help to write a program that computes the distance, defined as the number of cells in a shortest path, between any pair of cells.

Input

The input consists of several lines, each containing two integers a and b ($a, b \leq 10000$), denoting numbers of cells. The integers are always positive, except in the last line where $a = b = 0$ holds. This last line terminates the input and should not be processed.

Output

For each pair of numbers (a, b) in the input file, output the distance between the cells labeled a and b. The distance is the minimum number of moves to get from a to b.

Sample Input

```
19 30
0 0
```

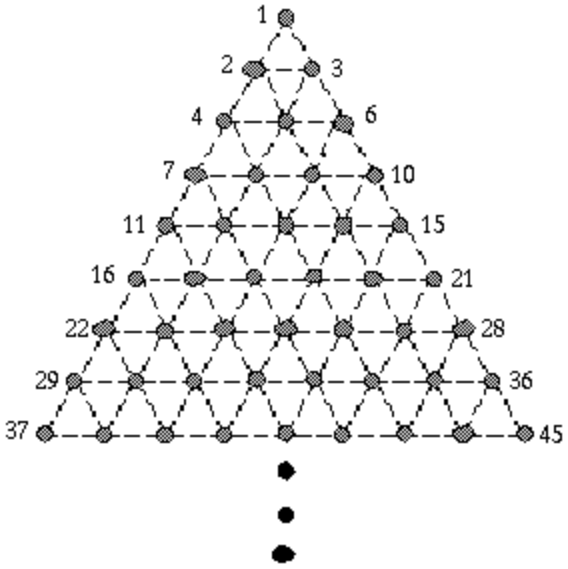
Sample Output

```
The distance between cells 19 and 30 is 5.
```

209 - Triangular Vertices

Time limit: 3.000 seconds

Consider the points on an infinite grid of equilateral triangles as shown below:



Note that if we number the points from left to right and top to bottom, then groups of these points form the vertices of certain geometric shapes. For example, the sets of points {1,2,3} and {7,9,18} are the vertices of triangles, the sets {11,13,26,24} and {2,7,9,18} are the vertices of parallelograms, and the sets {4,5,9,13,12,7} and {8,10,17,21,32,34} are the vertices of hexagons.

Write a program which will repeatedly accept a set of points on this triangular grid, analyze it, and determine whether the points are the vertices of one of the following ``acceptable'' figures: triangle, parallelogram, or hexagon. In order for a figure to be acceptable, it must meet the following two conditions:

1) Each side of the figure must coincide with an edge in the grid.

and 2) All sides of the figure must be of the same length.

Input

The input will consist of an unknown number of point sets. Each point set will appear on a separate line in the file. There are at most six points in a set and the points are limited to the range 1..32767.

Output

For each point set in the input file, your program should deduce from the number of points in the set which geometric figure the set potentially represents; e.g., six points can only represent a hexagon, etc. The output must be a series of lines listing each point set followed by the results of your analysis.

Sample Input

```
1 2 3
11 13 29 31
26 11 13 24
4 5 9 13 12 7
1 2 3 4 5
47
11 13 23 25
```

Sample Output

```
1 2 3 are the vertices of a triangle
11 13 29 31 are not the vertices of an acceptable figure
26 11 13 24 are the vertices of a parallelogram
4 5 9 13 12 7 are the vertices of a hexagon
1 2 3 4 5 are not the vertices of an acceptable figure
47 are not the vertices of an acceptable figure
11 13 23 25 are not the vertices of an acceptable figure
```

10245 - The Closest Pair Problem

Time limit: 3.000 seconds

Given a set of points in a two dimensional space, you will have to find the distance between the closest two points.

Input

The input file contains several sets of input. Each set of input starts with an integer **N (0<=N<=10000)**, which denotes the number of points in this set. The next **N** line contains the coordinates of **N** two-dimensional points. The first of the two numbers denotes the **X-coordinate** and the latter denotes the **Y-coordinate**. The input is terminated by a set whose **N=0**. This set should not be processed. The value of the coordinates will be less than **40000** and non-negative.

Output

For each set of input produce a single line of output containing a floating point number (with four digits after the decimal point) which denotes the distance between the closest two points. If there is no such two points in the input whose distance is less than **10000**, print the line **INFINITY**.

Sample Input

```
3
0 0
10000 10000
20000 20000
5
0 2
6 67
43 71
39 107
189 140
0
```

Sample Output

```
INFINITY
36.2215
```

805 - Polygon Intersections

Time limit: 3.000 seconds

Dark Blader has returned with new tactics. They are now able to create an energy field around its blade and if any blade enters inside any of these energy fields the energy level of the Bit-beast drastically decreases. So Tyson had to avoid these energy fields and finally he has won!

I was lucky enough to be around Kenny who was analyzing the game with his PC and helping Tyson to avoid the energy field. I saw that the energy field was Square in shape and the blade was at its centre. At that instant a problem came to my mind and let me see how efficiently you can solve that problem.

There will be N points in a 2D plane. Find out the maximum size such that if you draw such size squares around each point (that point will be at the center of the square) no two squares will intersect each other (can touch but not intersect). To make the problem simple the sides of the squares will be parallel to X and Y axis.

Input:

Input contains several test cases. Each case starts with N which will be at most 10,000 except one case which will be 100,000. Then there are N lines- pairs of integers denoting the coordinate of each point. The absolute value of the integers can be at most 1,000,000. X or Y coordinate of any two points will be unequal.

Output:

Output a single line for each test case- maximum side length of square.

SAMPLE INPUT	OUTPUT FOR SAMPLE INPUT
2 0 0 2 2	2

11378 - Bey Battle

Time limit: 5.000 seconds

Dark Blader has returned with new tactics. They are now able to create an energy field around its blade and if any blade enters inside any of these energy fields the energy level of the Bit-beast drastically decreases. So Tyson had to avoid these energy fields and finally he has won!

I was lucky enough to be around Kenny who was analyzing the game with his PC and helping Tyson to avoid the energy field. I saw that the energy field was Square in shape and the blade was at its centre. At that instant a problem came to my mind and let me see how efficiently you can solve that problem.

There will be N points in a 2D plane. Find out the maximum size such that if you draw such size squares around each point (that point will be at the center of the square) no two squares will intersect each other (can touch but not intersect). To make the problem simple the sides of the squares will be parallel to X and Y axis.

Input:

Input contains several test cases. Each case starts with N which will be at most 10,000 except one case which will be 100,000. Then there are N lines- pairs of integers denoting the coordinate of each point. The absolute value of the integers can be at most 1,000,000. X or Y coordinate of any two points will be unequal.

Output:

Output a single line for each test case- maximum side length of square.

SAMPLE INPUT	OUTPUT FOR SAMPLE INPUT
2 0 0 2 2	2

10180 - Rope Crisis in Ropeland!

Time limit: 3.000 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.

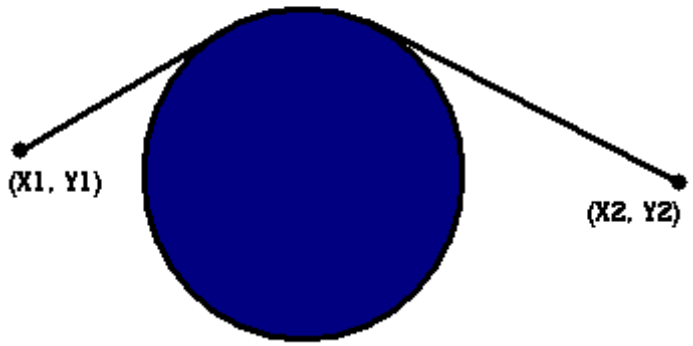


Fig 1: Situation when two groups have the round pillar between them. The pulled rope is never a straight line

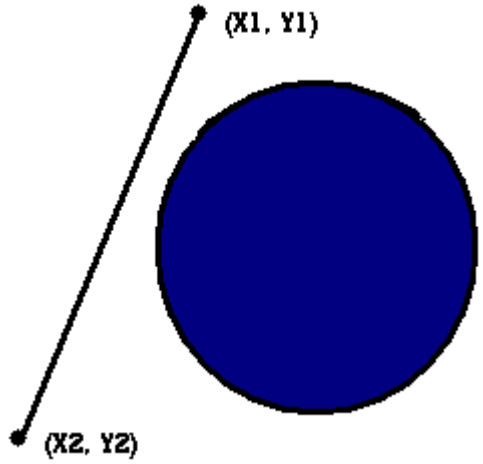


Fig 2: Situation when the two groups don't have the round pillar between them

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
```

10026 - Shoemaker's Problem

Time limit: 3.000 seconds

Shoemaker has N jobs (orders from customers) which he must make. Shoemaker can work on only one job in each day. For each i_{th} job, it is known the integer T_i ($1<=T_i<=1000$), the time in days it takes the shoemaker to finish the job. For each day of delay before starting to work for the i_{th} job, shoemaker must pay a fine of S_i ($1<=S_i<=10000$) cents. Your task is to help the shoemaker, writing a programm to find the sequence of jobs with minimal total fine.

The Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

First line of input contains an integer N ($1<=N<=1000$). The next N lines each contain two numbers: the time and fine of each task in order.

The Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

You programm should print the sequence of jobs with minimal fine. Each job should be represented by its number in input. All integers should be placed on only one output line and separated by one space. If multiple solutions are possible, print the first lexicographically.

Sample Input

```
1

4
3 4
1 1000
2 2
5 5
```

Sample Output

```
2 1 3 4
```

10161 - Ant on a Chessboard

Time limit: 3.000 seconds

Background

One day, an ant called Alice came to an M*M chessboard. She wanted to go around all the grids. So she began to walk along the chessboard according to this way: (you can assume that her speed is one grid per second)

At the first second, Alice was standing at (1,1). Firstly she went up for a grid, then a grid to the right, a grid downward. After that, she went a grid to the right, then two grids upward, and then two grids to the left…in a word, the path was like a snake.

For example, her first 25 seconds went like this:

(the numbers in the grids stands for the time when she went into the grids)

25	24	23	22	21
10	11	12	13	20
9	8	7	14	19
2	3	6	15	18
1	4	5	16	17

5
4
3
2
1

1 2 3 4 5

At the 8th second , she was at (2,3), and at 20th second, she was at (5,4).
Your task is to decide where she was at a given time.
(you can assume that M is large enough)

Input

Input file will contain several lines, and each line contains a number N(1<=N<=2*10^9), which stands for the time. The file will be ended with a line that contains a number 0.

Output

For each input situation you should print a line with two numbers (x, y), the column and the row number, there must be only a space between them.

Sample Input

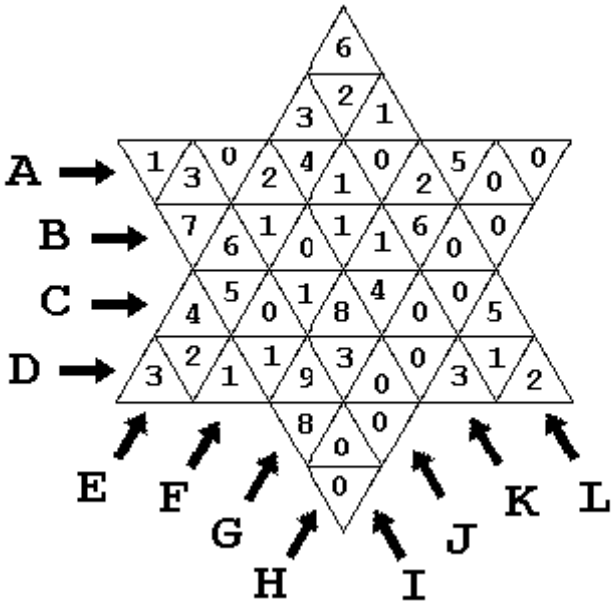
```
8
20
25
0
```

Sample Output

```
2 3
5 4
1 5
```

10159 - Star

A board contains 48 triangular cells. In each cell, there is written a digit (in a range from 0 through 9). Every cell belongs to two or three lines. These lines are marked by letters from A through L.



An example is depicted on the Figure. There, the cell containing digit 9, belongs to lines D, G and I. The cell containing digit 7, belongs to lines B and I.

For each line, A, B, C, ..., L, we consider the largest digit lying on it. In the above example, the largest digit for line A is 5, for line B is 7, for line E is 6, for line H is 0, for line J is 8 and so on.

Write a program, that inputs the largest digit for any one of the depicted 12 lines. The program should find out the smallest and the largest possible sum of digits located in all the cells of the board.

Input

Every line in the input contains 12 digits, each two of them separated by a space. The first of these digits means the largest one in line A, the second means the largest one in line B, and so on, the last digit means the largest one in line L.

Example

5 7 8 9 6 1 9 0 9 8 4 6

Output

For every line in the input file write the value of the smallest and of the largest possible sum of digits located in the cells of the board, on a single line. These two values should be separated by one space exactly. If there does not exists a solution, your program must outputs the words NO SOLUTION instead of the above two values.

Example

40 172

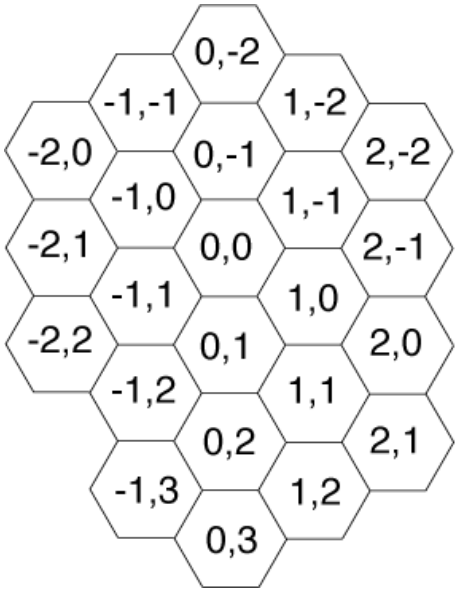
10182 - Bee Maja

Time limit: 3.000 seconds

Maja is a bee. She lives in a bee hive with thousands of other bees. This bee hive consists of many hexagonal honey combs where the honey is stored in.
But bee Maja has a problem. Willi told her where she can meet him, but because Willi is a male drone and Maja is a female worker they have different coordinate systems.

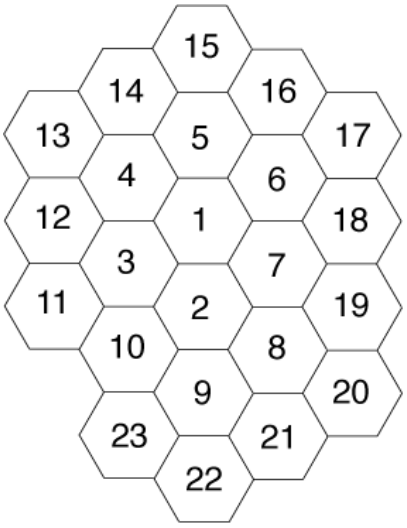
Maja's Coordinate System

Maja who often flies directly to a special honey comb has laid an advanced two dimensional grid over the whole hive.



Willi's Coordinate System

Willi who is more lazy and often walks around just numbered the cells clockwise starting from 1 in the middle of the hive.



Help Maja to convert Willi's system to hers. Write a program which for a given honey comb number gives the coordinates in Maja's system.

Input Specification

The input file contains one or more integers which represent Willi's numbers. Each number stands on its own in a separate line, directly followed by a newline. The honey comb numbers are all less than 100 000.

Output Specification

You should output the corresponding Maja coordinates to Willi's numbers, each coordinate pair on a separate line.

Sample Input

```
1
2
3
4
5
```

Sample Output

```
0 0
0 1
-1 1
-1 0
0 -1
```

10177 - (2/3/4)-D Sqr/Rects/Cubes/Boxes?

Time limit: 3.000 seconds

You can see a (4x4) grid below. Can you tell me how many squares and rectangles are hidden there? You can assume that squares are not rectangles. Perhaps one can count it by hand but can you count it for a (100x100) grid or a (10000x10000) grid. Can you do it for higher dimensions? That is can you count how many cubes or boxes of different size are there in a (10x10x10) sized cube or how many hyper-cubes or hyper-boxes of different size are there in a four-dimensional (5x5x5x5) sized hypercube. Remember that your program needs to be very efficient. You can assume that squares are not rectangles, cubes are not boxes and hyper-cubes are not hyper-boxes.

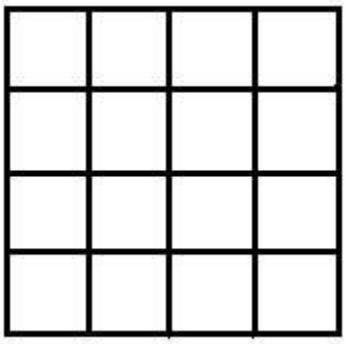


Fig: A 4x4 Grid

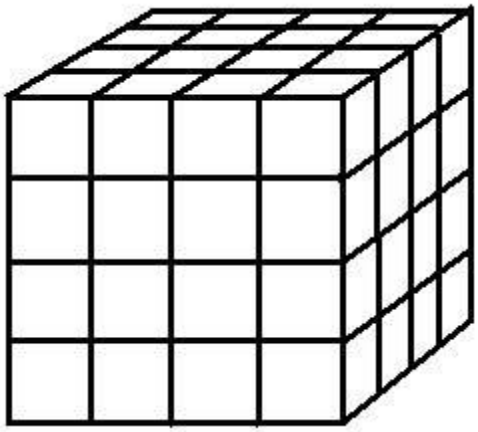


Fig: A 4x4x4 Cube

Input

The input contains one integer **N (0<=N<=100)** in each line, which is the length of one side of the grid or cube or hypercube. As for the example above the value of **N** is 4. There may be as many as 100 lines of input.

Output

For each line of input, output six integers **S2, R2, S3, R3, S4, R4** in a single line where **S2** means no of squares of different size in **(NxN)** two-dimensional grid, **R2** means no of rectangles of different size in **(NxN)** two-dimensional grid. **S3, R3, S4, R4** means similar cases in higher dimensions as described before.

Sample Input:

```
1
2
3
```

Sample Output:

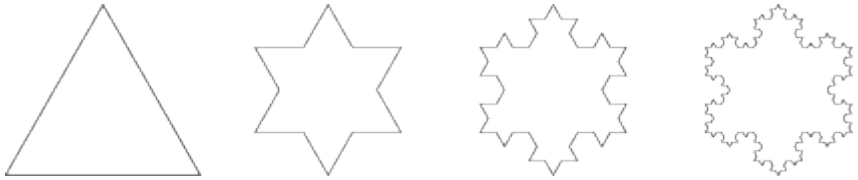
```
1 0 1 0 1 0
5 4 9 18 17 64
14 22 36 180 98 1198
```

10609 - Fractal

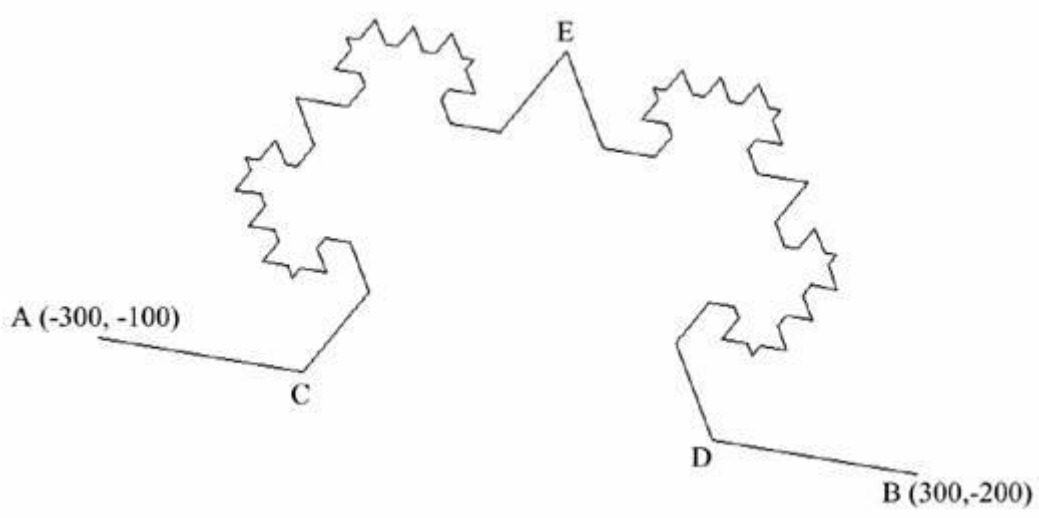
Time limit: 3.000 seconds

A fractal is a rough or fragmented geometric shape that can be subdivided in parts, each of which is (at least approximately) a smaller copy of the whole. Fractals are generally self-similar (bits look like the whole) and independent of scale (they look similar, no matter how close you zoom in).

Below you can see picture of a well-known fractal. Actually this picture shows the steps of the making of a fractal:



In this problem you will have to draw a fractal very similar to the one above. The fractal that you have to work with is given below:



In real life it is impossible to draw a fractal exactly according to its definition because somewhere we must stop drawing it. For example in the picture above we have stopped drawing when the length of the line on which a triangle has to be drawn is less than five.

Now let us discuss in detail how the fractal is to be drawn. You will be given the coordinates of **A (x_1, y_1)** and **B (x_2, y_2)**. In the figure above the coordinate of **A** is **(-300, -100)** and the coordinate of **B** is **(300, -200)**. **C** and **D** are the points, which divides **AB** in ratio **1:3** and **3:1**. So now you have to draw an equilateral triangle **CED** based on **CD**, of course the base is erased. And then you find two points, which divide **CE** in the ratio **1:3** and **3:1**. The same thing applies for **ED** and this process continues recursively up to the point when the length of the side of drawn equilateral triangles is less than a certain value **T**. Now if you look at the picture above you will find that it has two terminal points **A** and **B** and many corner points like **C**, **E** and **D**. Your job is to find the coordinates of these terminal points and corner points and print them in a certain order.

Input

The input file contains less than **10** lines of input.

Each line contains five numbers. The first four numbers are the coordinates x_1, y_1, x_2, y_2 **(-10000<x_1,y_1,x_2,y_2<10000)** and the last number **T(1<T<1000)** is the terminating threshold value. I mean when the line to be drawn will be less than **T** drawing will stop. The value of **T** will be such that the length of the line to be drawn will never be equal to **T**.

Input is terminated by a case whose value of **T** is less than **1**. This case should not be processed.

Output

For each line of input you should output **S+2** lines of outputs. The first line is the serial no of the output as shown in the sample output. Next line contains the number **S**, where **S** is the number of vertex and terminal points in the drawn fractal. Each of the **S** lines after that contains two floating-point numbers indicating the coordinate of one terminal point or vertex. The terminal points should be sorted in increasing order of the value of abscissa of the coordinate. In case of a tie the points should be sorted in ascending order of the ordinate. Two values are considered same if they differ by a value less than **1e-8**. All printed floating point numbers have five digits after the decimal point. Errors less than $\mathbf{2*10^{-5}}$ will be tolerated.

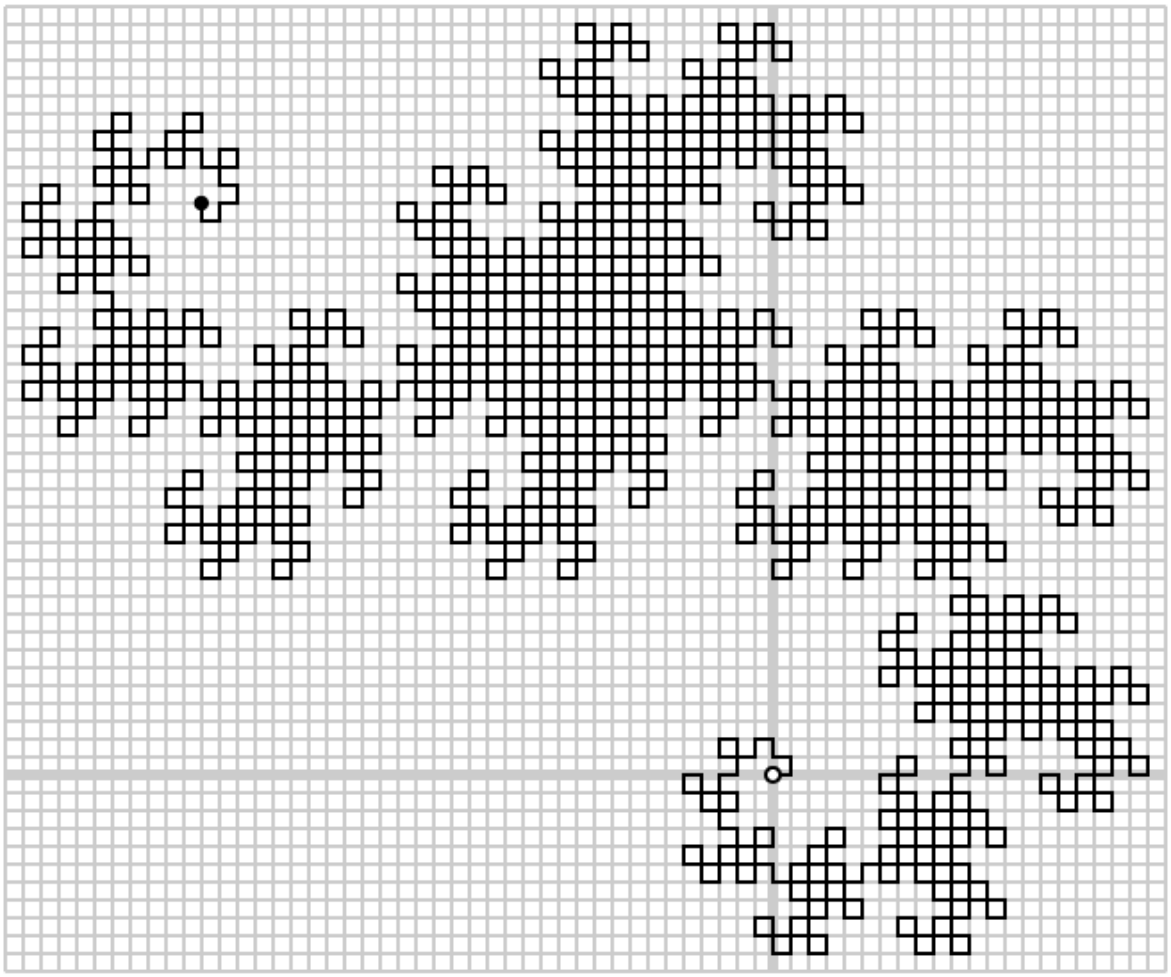
Sample Input	**Output for Sample Input**
10 10 -10 -10 5.1 -10 -10 10 10 5.1 5 5 -5 -8 .3	Case 1: 11 -10.00000 -10.00000 -5.00000 -5.00000 -1.58494 -5.91506 0.24519 -12.74519 5.00000 5.00000 5.24519 -7.74519 5.91506 1.58494 7.74519 -5.24519 8.66025 -8.66025 10.00000 10.00000 12.74519 -0.24519 Case 2: 11 -12.74519 0.24519 -10.00000 -10.00000 -8.66025 8.66025 -7.74519 5.24519 -5.91506 -1.58494 -5.24519 7.74519 -5.00000 -5.00000 -0.24519 12.74519 1.58494 5.91506 5.00000 5.00000 10.00000 10.00000

11253 - Fractal

Time limit: 3.000 seconds

Define $r(s)$ to be the complement of the reverse of the binary string s. i.e. Reverse s and then convert all 1's to 0's and all 0's to 1's. Further define a sequence of binary string as follow: s_0 = 1 and $s_n = s_{n-1}1r(s_{n-1})$. i.e.
s_0 = 1
s_1 = 1**1**0
s_2 = 110**1**100
s_3 = 1101100**1**1100100
...

We then program a robot to move at a steady speed of 1 unit per second and make a right-angle turn according to the characters of s_{10} after every unit of movement. At the k^{th} turn, the robot turns to left if the k^{th} character of s_{10} is a 1, and to right otherwise. The figure below shows the whole path of the robot.



The robot is placed at the origin (the small circle) and face east originally. It ends up at the coordinates (-32,32) (the small spot) after 2048 seconds. The path of the robot is known as a

dragon curve, a pretty well-known pattern of fractal.

If the robot is now programmed with input string s_{30} (with identical initial conditions as above), it will keep moving and then stop after 2^{31} seconds. We want to know the location of the robot at any given time.

Input
Input consists of multiple problem instances. Each instance consists of a single non-negative integer *n*, where $n \leq 10^9$. The input data is terminated by a negative integer. There will be less than 5000 test cases.

Output
For each input integer *n*, print out the location of the robot right after *n* second since the robot starts its journey with input string s_{30}. The location should be printed with the format "`(x,y)`" in a single line.

Sample input
```
1
2
3
2048
1000000000
-1
```

Sample output
```
(1,0)
(1,1)
(0,1)
(-32,32)
(9648,-31504)
```