XXIII Maratón Nacional de Programación
ACIS REDIS 2009
ACM ICPC

Problemas

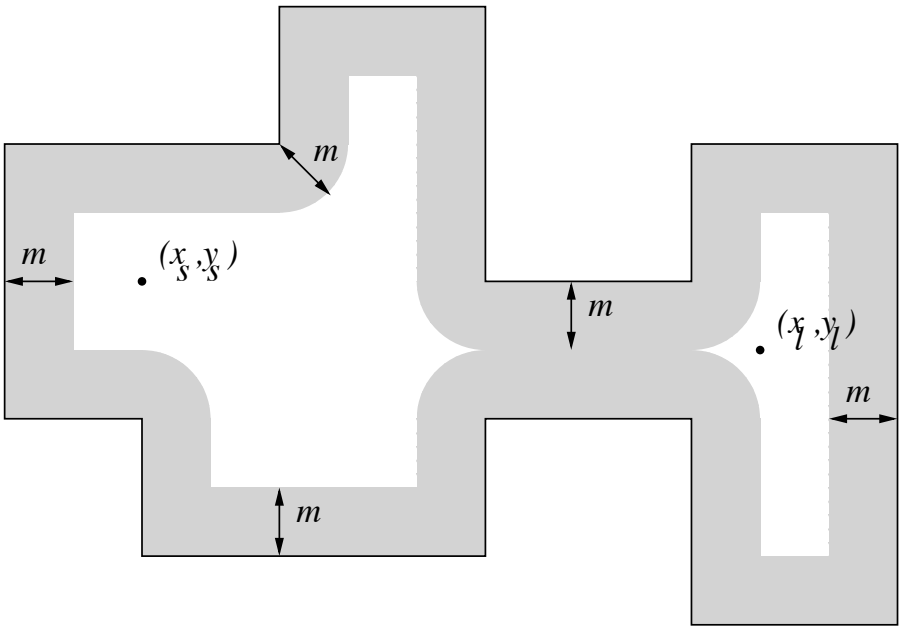
(Este conjunto contiene 10 problemas; páginas numeradas de 1 a 19)

Problem A
Message in the Enemy Territory

Source file name: `message.c`, `message.cpp` *or* `message.java`

A group of commandos has been caught and sent to a maximum-security prison in enemy territory. In order to escape from the prison, a soldier needs to give a message to the squadron leader.

The boundary of the prison is protected by electronic alarms: for his security, the soldier needs to keep a distance greater than m from the boundary. An additional restriction is that the soldier can only stand on those positions with integer coordinates. In each step, the soldier can move, from a given position (x, y), only to the nearby positions: $(x-1, y-1)$, $(x-1, y)$, $(x-1, y+1)$, $(x, y-1)$, $(x, y+1)$, $(x+1, y-1)$, $(x+1, y)$ and $(x+1, y+1)$, without going out of the interior of the prison. The walls of the prison form a simple polygon (no repeated vertices and no intersections between edges) and all of them are parallel to either the x-axis or the y-axis of a hypothetical coordinate system. The following figure shows a typical prison's plan:



(x_s, y_s) and (x_l, y_l) corresponds to the position of the soldier and the squadron leader respectively. The gray area indicates those positions that are at distance less than or equal to m from the prison's boundary, i.e., the zone that the soldier cannot stand on.

A *safe* path is a sequence of pairs of integer coordinates, each one at a distance greater than m from the boundary of the prison, so that consecutive pairs are different and do not differ in more than one in each coordinate. In the depicted example, there is not a safe path from the soldier to the squadron leader.

Your task is to determine, for a given prison's plan, if there exists a safe path from the soldier position to the squadron leader position.

Input

The problem input consists of several test cases. Each test case consists of three lines:

- The first line contains two integer numbers separated by blanks, n and m, with $4 \leq n \leq 1000$ and $1 \leq m \leq 30$, indicating the number of the prison's boundary vertices and the alarm range respectively.
- The second line contains a list of $2 \cdot n$ integer numbers, $x_1, y_1, \ldots, x_n, y_n$, separated by blanks: the list of vertices of a simple n-polygon that describes the boundary of the prison. $0 \leq x_i, y_i \leq 1000$.
- The last line contains four integer numbers separated by blanks, x_s, y_s, x_l, and y_l, indicating the position of the soldier and the position of the squadron leader ($0 \leq x_s, y_s \leq 1000$, $0 \leq x_l, y_l \leq 1000$).

The end of the input is indicated by a line with "`0 0`".

The input must be read from the file message.in.

Output

For each test case the output includes a line with the word "`Yes`" if there exists a path from the soldier to the squadron leader. Otherwise the word "`No`" must be printed.

The output must be written to standard output.

Sample input	**Output for the sample input**

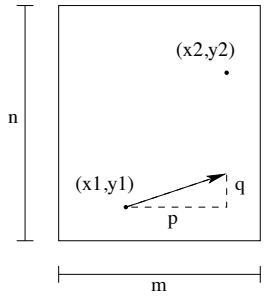
`4 1`
`0 0 0 5 5 5 5 0`
`2 2 3 3`
`8 3`
`0 16 0 6 4 6 4 0 12 0 12 10 8 10 8 16`
`4 12 8 4`
`0 0`
	`Yes`
	`No`

Problem B
Rational Billiard

Source file name: `billiard.c, billiard.cpp` *or* `billiard.java`

The International Billiard Manufactory (IBM) builds the best billiard tables in the world. Its last product is the Rational Billiard, which is a friction-less table with volume-less balls. This means that balls just occupy a point in the space, and once they are struck with the cue, they move on the table with constant velocity, until they hit another ball.

IBM wants you to build a program to analyse the behavior of balls in the Rational Billiard. The program has to decide whether a ball, struck with certain angle, will hit another ball in certain position. The next figure shows the configuration of the table and the balls:



m and n are integer values that indicate the size of the table. (x_1, y_1) and (x_2, y_2) are the coordinates of the first and second ball respectivelly, with $0<x_1, x_2<m$ and $0<y_1, y_2<n$. The direction in which the first ball is struck is determined by the integer values p and q. More precisely, the slope $\frac{q}{p}$ determines the hitting direction of the cue, where p and q cannot be zero simultaneously. A value of $p=0$ means that the ball moves parallel to the vertical axis, according to the sign of q. When the ball hits an edge, it rebounds like if the edge were a mirror (incidence angle = reflection angle). In the special case when the ball hits a corner, it is reflected on the same line but in the opposite direction of arrival. Finally, spin effects are neglected, too.

Input

Each line in the input corresponds to a test case specified by eight integer values: $m, n, x_1, y_1, x_2, y_2, p$ and q, with $0<m, n\leq 1000$, $0<x_1, x_2<m$, $0<y_1, y_2<n$, $-1000\leq p, q\leq 1000$, and $|p|+|q|>0$.

The end of the input is specified by a line with the string "`0 0 0 0 0 0 0 0`".

The input must be read from the file billiard.in.

Output

For each test case, the program must output a line with the the text "`HIT`" if the first ball hits the second ball, or "`MISS`" if it does not.

The output must be written to standard output.

Sample Input	**Sample output**

`4 4 3 1 1 1 1 1`
`4 4 3 1 2 2 1 1`
`0 0 0 0 0 0 0 0`
	`HIT`
	`MISS`

Problem C
Best Coalitions

Source file name: `coalit.c`, `coalit.cpp` *or* `coalit.java`

Envy Inc. is a joint stock company, that is, a company in which every stockholder legally owns shares of stock that account for some percentage of the total shares of stock of the company. Due to the global economic crisis, the management rules of Envy Inc. define a particular way for distributing last year's profit: if a stockholder owns more than half of the shares of stock, he/she wins the total profit. Nothing fancy so far in this wild world!

Nevertheless, there are situations in which no stockholder owns more than 50% of the shares of stock of the company. So, in order to gain some profit, stockholders are allowed to form *coalitions*, i.e., groups of stockholders. The participation of the coalition, share-wise, is equivalent to the sum of its stockholders' percentile participation. Hence, if a coalition has more than half of the shares of stock, its members win the totality of last years profit. Then, the members of the coalition receive a part of the profit proportional to their individual participation in the coalition.

For instance, let us assume there are 5 stockholders: A, B, C, D and E, owning 20%, 12%, 14%, 29% and 25% of the stock of the company, respectively. The stockholder E could form several winning coalitions. For example, if E were to form a coalition with A and B, he/she would get 43.86% of last year's profit. If E were to form a coalition with B and C instead, he/she would get 49.02% of last year's profit. On the other hand, E could not form a winning coalition with only A.

Your problem is, given a distribution of shares of stock of Envy Inc., and a stockholder, to determine the maximum percentage of the last year's profit that the given stockholder may win.

Input

The input consists of several test cases, each one defining a percentile distribution of shares of stock, and the index of a stockholder to determine his/her optimal participation. More precisely, each test case is defined by several input lines:

- the first line contains two integer values n ($1 \leq n \leq 100$) and x ($1 \leq x \leq n$), separated by a blank, representing the number of stockholders in Envy Inc. and the index of a stockholder to determine his/her optimal participation, respectively;
- each one of the following n lines has a single floating point value p_i, rounded to 2 decimal places, which represents the percentage of stock ownership of stockholder i ($1 \leq i \leq n$). The floating point delimiter is '.' (i.e. the dot). You can assume that $p_1 + \ldots + p_n = 100$.

The end of the input is indicated by $n = x = 0$, an artificial case that must be ignored.

The input must be read from the file coalit.in.

Output

For each given case, output a single line with the corresponding answer. The answer should be formatted and approximated to two decimal places. The floating point delimiter must be '.' (i.e. the dot). The rounding applies towards the *nearest neighbor* unless both neighbors are equidistant, in which case the result is rounded up (e.g. 78.312 is rounded to 78.31; 78.566 is rounded to 78.57; 78.345 is rounded to 78.35, etc.).

The output must be written to standard output.

Sample input	**Output for the sample input**
5 5 20.00 12.00 29.00 14.00 25.00 2 1 56.87 43.13 2 2 56.87 43.13 3 1 10.00 45.00 45.00 0 0	49.02 100.00 43.13 18.18

Problem D
Informants

Source file name: `inform.c`, `inform.cpp` *or* `inform.java`

The Agency of Counter-intelligence of Macondo, ACM in short, is willing to put to the test the quality of its informants with a simple, yet highly accurate, procedure that evaluates how trustworthy a group of informants is.

The ACM determines the confidence of a group of informants by surveying the confidence among them: informants assert their particular opinions about other ones, even themselves. As a result of the survey, the ACM gathers a set of assertions of the form "X says Y is reliable" or "X says Y is not reliable". If X happens to be reliable, the ACM assumes that whatever he or she says, can be interpreted to be true. Otherwise, if X is not reliable, his or her opinions may be either true or false. At the end, the ACM qualifies the situation by determining the maximum number of informants that can be reliable according to the surveyed answers.

As an example, let's assume there are four informants A, B, C and D, with the following surveyed answers: "A says B is reliable but D is not", "B says C is not reliable", and "C says A and D are reliable". In this case, it happens that at most two informants are reliable.

Your task is to help the ACM by writing an efficient program that, given the results of the survey, computes the maximum number of informants that may be reliable.

Input

The problem's input has several cases. Each test case begins with a line with two nonnegative integer numbers, i ($0<i\leq 20$) and a ($0\leq a\leq 800$), separated by blanks. i is the number of informants and a is the number of answers in the confidence survey. Then, a lines follow, each one with two integer numbers x and y ($1\leq x\leq i$, $1\leq |y|\leq i$), separated by blanks. If y is positive, the input line means that "informant x says informant y is reliable". If y is negative, the then the line means that "informant x says informant y is not reliable". The end of the input is indicated by a line with two 0 values (an artificial case that should be ignored).

The input must be read from the file inform.in.

Output

For each input case, output in a single line the corresponding answer, i.e., the maximum number of reliable informants according to the answers in the survey.

The output must be written to standard output.

Sample input	**Output for the sample input**
4 5 1 2 1 -4 2 -3 3 1 3 4 1 1 1 -1 3 3 1 2 2 3 3 -1 0 0	2 0 2

Problem E
Look-and-Say sequences

Source file name: `lookandsay.c, lookandsay.cpp` *or* `lookandsay.java`

A look-and-say sequence is a sequence of integers, expressed in decimal notation, where each sucessive term is generated by *describing* the previous one.

For instance, if x_1 (the first term of the sequence) is 1, the next term is the description of this term, 11 ("one 1"), which is described by 21 ("two 1's"), which is described by 1211 ("one 2 one 1"), etc.; the series continues 111221, 312211, 13112221, ...

Your problem is to build a program that, given the first term of a look-and-say sequence x_1, calculates the j-th digit of the i-th term, x_i.

Input

Each line in the input corresponds to a test case specified by 3 integer values: x_1, i and j, with $1 \leq x_1 \leq 1000$, $1 \leq i \leq 1000$ and $1 \leq j \leq \min(\lfloor \log_{10}(x_i)+1 \rfloor, 1000)$. The end of the input is indicated by a line "`0 0 0`".

The input must be read from the file lookandsay.in.

Output

For each test case, the program must output a line with the j-th digit of the term x_i of the look-and-say sequence that starts with the term x_1.

The output must be written to standard output.

Sample Input	**Sample output**

`1 3 1`
`1 3 2`
`1 7 2`
`123 3 1`
`0 0 0`
	`2`
	`1`
	`3`
	`3`

Problem F
Burger Time?

Source file name: `burger.c, burger.cpp` *or* `burger.java`

Everybody knows that along the more important highways there are countless fast food restaurants. One can find easily hamburgers, hot dogs, pizzas, sandwiches ... food everywhere.

Many times the problem isn't to find a restaurant but a drugstore. After a big lunch with fast food it's normal that we need a drugstore because our stomach begins to feel pain.

Given the locations of the restaurants and drugstores on a highway, you want to determine the minimum distance between a restaurant and a drugstore.

Input

The first line of each test case gives an integer L ($1 \leq L \leq 2000000$) indicating the length of the highway.

The second line of each test case contains a string S of length L, showing the positions of the restaurants and drugstores along the highway in the following manner:

- The character "`R`" represents a place with a restaurant.
- The character "`D`" represents a place with a drugstore.
- The character "`Z`" represents a place with a restaurant and a drugstore.
- The character "`.`" represents an empty place.

You can suppose that every test case has at least one restaurant and at least one drugstore.

The end of the input is indicated when $L = 0$.

The input must be read from the file burger.in.

Output

For each case in the input, print one line with the minimum distance between a restaurant and a drugstore.

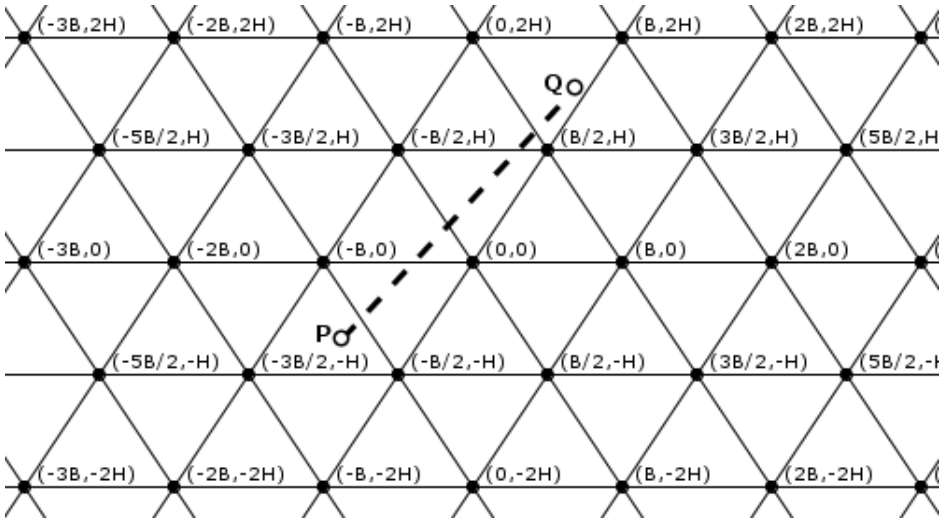
The output must be written to standard output.

Sample Input	**Sample output**
2 RD 5 ..Z.. 10 .R......D. 10 .R..Z...D. 10 ...D..R... 25 ..D...R.RR...DD...D.R...R 0	1 0 7 0 3 2

Problem G
Triangular Grid

Source file name: `grid.c`, `grid.cpp` *or* `grid.java`

There is an infinite grid in the Cartesian plane composed of isosceles triangles, with the following design:



A *single triangle* in this grid is a triangle with vertices on intersections of grid lines that has not other triangles inside it.

Given two points P and Q in the Cartesian plane you must determine how many single triangles are intersected by the segment $\overline{PQ}$. A segment intersects a polygon if and only if there exists one point of the segment that lies inside the polygon (excluding its boundary).

Note that the segment $\overline{PQ}$ in the example intersects exactly six single triangles.

Input

The problem input consists of several cases, each one defined in a line that contains six integer values B, H, x_1, y_1, x_2 and y_2 ($1 \leq B \leq 200,\ 2 \leq H \leq 200,\ -1000 \leq x_1, y_1, x_2, y_2 \leq 1000$), where:

- B is the length of the base of all isosceles single triangles of the grid.
- H is the height of all isosceles single triangles of the grid.
- (x_1, y_1) is the point P, that defines the first extreme of the segment.
- (x_2, y_2) is the point Q, that defines the second extreme of the segment.

You can suppose that neither P nor Q lie in the boundary of any single triangle, and that $P \neq Q$.

The end of the input is specified by a line with the string "`0 0 0 0 0 0`".

The input must be read from the file grid.in.

Output

For each case in the input, print one line with the number of single triangles on the grid that are intersected by the segment $\overline{PQ}$.

The output must be written to standard output.

Sample Input	**Sample output**

`100 120 -20 -100 160 160`
`10 8 5 5 5 4`
`10 8 5 5 10 5`
`10 8 5 5 10 10`
`0 0 0 0 0 0`
	6
	1
	2
	3

Problem H
GrayInc

Source file name: `gray.c`, `gray.cpp` *or* `gray.java`

Gray Inc. is a software fabricant specialized in management of Gray codes. An n-binary code, for $n \geq 1$, is a sequence of words $w_0, w_1, ...$ that includes every possible binary word of n bits. An n-Gray code is an n-binary code such that between any two consecutive words there is only one bit that changes, i.e., they differ at exactly one position. As you might be aware by now, there are many n-Gray codes.

Gray Inc. produces its own particular kind of Gray code in the following way (name G_n the produced n-Gray code, $n \geq 1$):

$$G_n = \begin{cases} \langle 0, 1 \rangle & \text{if } n = 1 \\ (0G_{n-1})\,(1G_{n-1}^R) & \text{if } n \geq 2 \end{cases}$$

The notation defining G_n should be understood as follows:

- bA : appends bit b to every element of the sequence A;
- AB : concatenates sequences A and B;
- A^R : sequence with the elements of sequence A in reversed order.

For instance,

$$\begin{aligned} G_2 &= (0G_1)\,(1G_1^R) \\ &= (0\langle 0, 1\rangle)\,(1\langle 0, 1\rangle^R) \\ &= (0\langle 0, 1\rangle)\,(1\langle 1, 0\rangle) \\ &= \langle 00, 01\rangle\langle 11, 10\rangle \\ &= \langle 00, 01, 11, 10\rangle \end{aligned}$$

and

$$\begin{aligned} G_3 &= (0G_2)\,(1G_2^R) \\ &= (0\langle 00, 01, 11, 10\rangle)\,(1\langle 00, 01, 11, 10\rangle^R) \\ &= (0\langle 00, 01, 11, 10\rangle)\,(1\langle 10, 11, 01, 00\rangle) \\ &= \langle 000, 001, 011, 010\rangle\langle 110, 111, 101, 100\rangle \\ &= \langle 000, 001, 011, 010, 110, 111, 101, 100\rangle \end{aligned}$$

Observe that not only G_n is an n-Gray code, but also a *circular* Gray code as the first word in the sequence may be regarded as the successor of the last one in the sequence.

Gray Inc. wants your help to solve the following problem: given a binary word w in G_n and a natural number m, they want to produce the binary word in the sequence G_n that is m words ahead w in the listing (of course, considering the circular ordering described above). Can you help them?

Input

The problem input consists of several cases, each one defined by a line with an integer m ($0<m\leq 1000$), and a binary n-word w ($1\leq n\leq 100$), separated by one blank.

The end of the input is given by a line with m=0 and w=0.

The input must be read from the file gray.in.

Output

For each input case, your solution should output the n-binary word that is m words ahead of w in the listing of G_n.

The output must be written to standard output.

Sample input	**Output for the sample input**
1 0 3 0 1 1 1 11 6 011 123 010101010 0 0	1 1 0 10 000 111100100

Problem I
Langton's Ant

Source file name: `langton.c, langton.cpp` *or* `langton.java`

Langton's ant, after the mathematician Christopher Langton, is a cellular automaton with a very simple set of rules but interesting emergent behavior; this behavior is currently the matter of research for some mathematicians. The analogy between ants and Langton's cellular automaton comes from the observation that one can arbitrarily identify the state of the automaton as the "ant", and the dynamics of the automaton with the ability of the ant to travel in a special world.

The ant world's is an $n \times n$ plane where the squares (or cells) on the plane are colored variously either blue or red. The number n is called the *size* of the world. A cell is denoted with a pair (i, j), $(1 \leq i, j \leq n)$. The ant lives and moves in single steps following the rules below:

- if it is on a blue cell, it flips the color of the cell, turns $\frac{\pi}{2}$ to the left, and moves forward to the next cell in the direction it is facing;
- if it is on a red cell, it flips the color of the cell, turns $\frac{\pi}{2}$ to the right, and moves forward to the next cell in the direction it is facing; and
- if a movement is impossible (because the ant cannot move out of the world), then the ant dies.

For example, let us assume the ant is in a red cell while facing east. If there is not a cell immediately to its south, then the ant dies. On the contrary, if there is a cell immediately to its south, then the ant takes a single step by moving to this cell to which it arrives facing south and the color of its source cell flips to blue. Then, the ant will try to take another single step, and so on.

Your task is to determine if the ant can go to the (n, n) cell of a world, given (i) the configuration of the world, and (ii) the initial position of the ant. You are to assume the ant's initial direction is north.

Input

The configuration of an $n \times n$ world can be codified as a natural number in binary notation by using n^2 bits. We adopt the following conventions: 0=blue, 1=red, and the binary representation of the configuration identifies the cells of the world from left to right and from bottom to top (considering the bits from the most significant bit to the least significant bit). For example, the binary number 0100 (4 in decimal notation) represents a 2×2 world with the following configuration:

blue	blue
blue	red

Coherently, the binary number 011010100 (212 in decimal notation) represents a 3×3 world with the following configuration:

red	blue	blue
blue	red	blue
blue	red	red

The problem input has several test cases. Each case consists of a single line containing a list of four natural numbers, n, c, x, y, separated by blanks, that should be interpreted as:

- n ($1 \leq n \leq 16$): the size of the world;
- c ($0 \leq c < 2^{(n^2)}$): decimal representation of an n^2-bit binary number that describes an initial configuration of the world, as above explained;
- (x, y): coordinates of the initial position of the ant in the world ($1 \leq x, y \leq n$), where the position (n, n) corresponds to the least significant bit of c.

The end of the input is indicated by a line where $n=c=x=y=0$.

The input must be read from the file langton.in.

Output

For each test case your solution should output:

- `Yes` if the ant reaches the cell (n, n) from the initial position;
- `Kaputt!` if the ant dies without reaching the cell (n, n) from the initial position.

For each test case, it's guaranteed that after a finite number of steps, the ant reaches the cell (n, n) or dies without reaching the cell (n, n).

The output must be written to standard output.

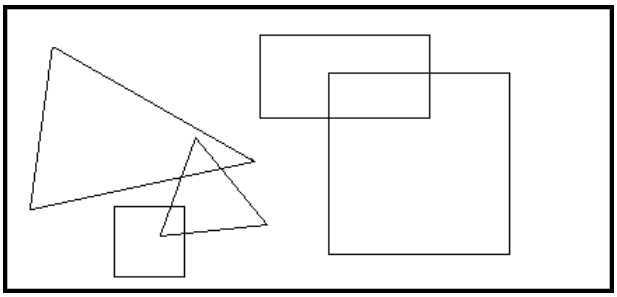
Sample input	**Output for the sample input**

`2 8 1 1`
`2 4 1 1`
`2 15 1 1`
`0 0 0 0`
	`Yes`
	`Kaputt!`
	`Kaputt!`

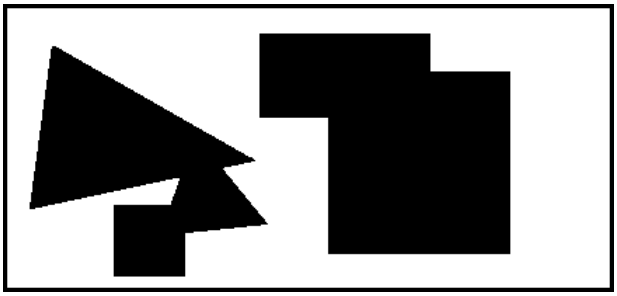
Problem J
Chinese Ink

Source file name: `ink.c`, `ink.cpp` *or* `ink.java`

Lucca, my four-years-old daughter, loves drawing polygons in bond paper. For example, yesterday she drew two squares, a rectangle and two triangles:



Today, she wanted to fill her figures with black chinese ink. I helped her, obtaining the next result:



She asked me: how many black zones do you see?. I said: two. I'm bored answering the same question everyday. Can you help us writing a program that, given a collection of black filled polygons, determines the number of black zones on the drawing?

For a precise understanding: a *black zone* is a region of black coloured points on the sheet, where every pair of them may be connected by a continuous line within the region.

Input

The input consists of several test cases. Each test case is represented as follows:

- A line with an integer N ($1 \leq N \leq 40$) which indicates the number of polygons in the drawing.

- N lines, one per polygon, each one containing a list of $2{\cdot}t$ integer numbers x_1 y_1 x_2 y_2 ... x_t y_t specifying the points in the boundary of the polygon ($-10^4 \leq x_i, y_i \leq 10^4$, $3 \leq t \leq 10$). The drawn polygon is bounded by the closed path composed of the straight line segments $\overline{(x_1, y_1)(x_2, y_2)}$, $\overline{(x_2, y_2)(x_3, y_3)}$, ..., $\overline{(x_{t-1}, y_{t-1})(x_t, y_t)}$, and $\overline{(x_t, y_t)(x_1, y_1)}$. You can suppose that the drawn polygon is a simple polygon (a polygon whose boundary is a non-self-intersecting closed path).

The end of the input is indicated when N=0.

The input must be read from the file ink.in.

Output

For each case in the input, print one line with the number of black zones in the drawing after filling each one of the polygons with black chinese ink.

The output must be written to standard output.

Sample Input	**Sample output**
5 35 29 179 111 19 145 183 22 305 22 305 80 183 80 232 49 361 49 361 178 232 178 137 94 188 156 112 164 79 144 129 143 129 193 79 193 2 20 20 30 20 30 30 20 30 40 40 40 50 50 50 50 40 2 20 20 30 20 30 30 20 30 30 30 40 30 40 40 30 40 3 20 20 40 20 40 40 20 40 50 30 60 20 70 50 60 40 50 30 30 50 0	2 2 1 1