UW-Madison 2009 ICPC Team Practice 2

Problem C
"Cable master"

Input file `cable.in`

Output file `cable.out`

Inhabitants of the Wonderland have decided to hold a regional programming contest. The Judging Committee has volunteered and has promised to organize the most honest contest ever. It was decided to connect computers for the contestants using a "star" topology - i.e. connect them all to a single central hub. To organize a truly honest contest, the Head of the Judging Committee has decreed to place all contestants evenly around the hub on an equal distance from it.

To buy network cables, the Judging Committee has contacted a local network solutions provider with a request to sell for them a specified number of cables with equal lengths. The Judging Committee wants the cables to be as long as possible to sit contestants as far from each other as possible.

The Cable Master of the company was assigned to the task. He knows the length of each cable in the stock up to a centimeter, and he can cut them with a centimeter precision being told the length of the pieces he must cut. However, this time, the length is not known and the Cable Master is completely puzzled.

You are to help the Cable Master, by writing a program that will determine the maximal possible length of a cable piece that can be cut from the cables in the stock, to get the specified number of pieces.

Input

The first line of the input file contains two integer numb ers N and K, separated by a space. N (1 = N = 10000) is the number of cables in the stock, and K (1 = K = 10000) is the number of requested pieces. The first line is followed by N lines with one number per line, that specify the length of each cable in the stock in meters. All cables are at least 1 meter and at most 100 kilometers in length. All lengths in the input file are written with a centimeter precision, with exactly two digits after a decimal point.

Output

Write to the output file the maximal length (in meters) of the pieces that Cable Master may cut from the cables in the stock to get the requested number of pieces. The number must be written with a centimeter precision, with exactly two digits after a decimal point.

If it is not possible to cut the requested number of pieces each one being at least one centimeter long, then the output file must contain the single number "0.00" (without quotes).

Sample input

```
4 11
8.02
7.43
4.57
5.39
```

Output for the sample input

```
2.00
```

Problem C: Exact Change

- Seller: That will be fourteen dollars.
- Buyer: Here's a twenty.
- Seller: Sorry, I don't have any change.
- Buyer: OK, here's a ten and a five. Keep the change.

When travelling to remote locations, it is often helpful to bring cash, in case you want to buy something from someone who does not accept credit or debit cards. It is also helpful to bring a variety of denominations in case the seller does not have change. Even so, you may not have the exact amount, and will have to pay a little bit more than full price. The same problem can arise even in urban locations, for example with vending machines that do not return change.

Of course, you would like to minimize the amount you pay (though you must pay at least as much as the value of the item). Moreover, while paying the minimum amount, you would like to minimize the number of coins or bills that you pay out.

Input Specification

The first line of input contains one integer specifying the number of test cases to follow. Each test case begins with a line containing an integer, the price in cents of the item you would like to buy. The price will not exceed 10 000 cents (i.e., $100). The following line contains a single integer *n*, the number of bills and coins that you have. The number *n* is at most 100. The following *n* lines each contain one integer, the value in cents of each bill or coin that you have. Note that the denominations can be any number of cents; they are not limited to the values of coins and bills that we usually use in Canada. However, no bill or coin will have a value greater than 10 000 cents ($100). The total value of your bills and coins will always be equal to or greater than the price of the item.

Sample Input

```
1
1400
3
500
1000
2000
```

Output Specification

For each test case, output a single line containing two integers: the total amount paid (in cents), and the total number of coins and bills used.

Output for Sample Input

```
1500 2
```

Ondřej Lhoták

Problem 2—Cylindrical Mirror
written by Andy Poe

Michelle Kwan is performing in *Skating With The Stars* with her very able partner, Rush Limbaugh. At the center of the rink is a large cylindrical mirror twenty feet in diameter. Michelle is looking into the mirror (from outside) to see if she can see Rush's reflection.

In Cartesian coordinates, you can imagine that the center of the mirror is at the origin and its radius is 10. Michelle is standing on the negative *x*-axis with (obviously) $x < -10$ and is facing the mirror. Rush is somewhere on the Cartesian plane (but *not* anywhere on the *x*-axis). It could very well be that Rush is in Michelle's direct line of sight, but for the skating stunt to work, Michelle has to locate Rush in the mirror. You are to compute the angle that Michelle has to look in order to see Rush's reflection, assuming it's even possible. For every case, Rush will either be visible in the mirror or not visible. No test case will have Rush on the "horizon" of the mirror.

INPUT SPECIFICATION. Each input case will be three floating-point numbers separated by one space and the case terminated by **<EOLN>**. The first floating point number is Michelle's location on the *x*-axis. The second is Rush's *x*-coordinate. The third is Rush's *y*-coordinate. The last input case will be followed by "0**<EOLN>**". This line is not to be processed; it merely specifies the end of input.

OUTPUT SPECIFICATION. The output cases should appear in the same order as the input cases. Each output case will be of the form "Case *c*: Michelle can see Rush's reflection *d* degrees to the *lr*." or "Case *c*: Michelle cannot see Rush's reflection." whichever is appropriate. *c* is the number of the input case, and *lr* is the word "left" or the word "right" whichever is appropriate. *d* is the appropriate angle rounded to the nearest tenth of a degree and displayed with exactly one digit to the right of the decimal point. Each output case should be followed by two **<EOLN>**'s.

SAMPLE INPUT.

```
-20·0·20<EOLN>
-20·10·10<EOLN>
0<EOLN>
<EOF>
```

SAMPLE OUTPUT.

```
Case·1:·Michelle·can·see·Rush's·reflection·28.7·degrees·to·the·left.<EOLN>
<EOLN>
Case·2:·Michelle·cannot·see·Rush's·reflection.<EOLN>
<EOLN>
<EOF>
```

Problem F
"The dog task"

Input file	DOG.IN
Output file	DOG.OUT
Time limit	10 seconds per test

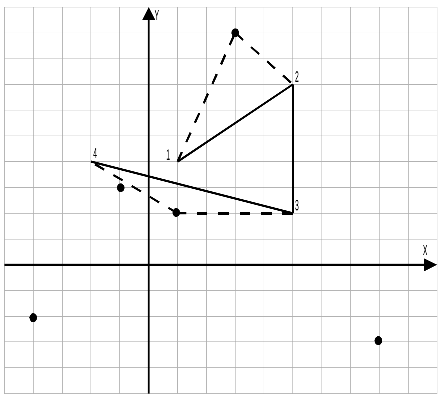
Hunter Bob often walks with his dog Ralph. Bob walks with a constant speed and his route is a polygonal line (possibly self-intersecting) whose vertices are specified by N pairs of integers (X_i, Y_i) – their Cartesian coordinates.

Ralph walks on his own way but always meets his master at the specified N points. The dog starts his journey simultaneously with Bob at the point (X_1, Y_1) and finishes it also simultaneously with Bob at the point (X_N, Y_N).

Ralph can travel at a speed that is up to two times greater than his master's speed. While Bob travels in a straight line from one point to another the cheerful dog seeks trees, bushes, hummocks and all other kinds of *interesting places* of the local landscape which are specified by M pairs of integers (X'_j, Y'_j). However, after leaving his master at the point (X_i, Y_i) (where $1 \le i < N$) the dog visits at most one interesting place before meeting his master again at the point (X_{i+1}, Y_{i+1}).

Your task is to find the dog's route, which meets the above requirements and allows him to visit the maximal possible number of interesting places. The answer should be presented as a polygonal line that represents Ralph's route. The vertices of this route should be all points (X_i, Y_i) and the maximal number of interesting places (X'_j, Y'_j). The latter should be visited (i.e. listed in the route description) at most once.

An example of Bob's route (solid line), a set of interesting places (dots) and one of the best Ralph's routes (dotted line) are presented in the following picture:



Input

The first line of the input file contains two integers N and M, separated by a space ($2 \le N \le 100$, $0 \le M \le 100$). The second line contains N pairs of integers X_1, Y_1, ..., X_N, Y_N, separated by spaces, that represent Bob's route. The third line contains M pairs of integers X'_1, Y'_1, ..., X'_M, Y'_M, separated by spaces, that represent interesting places.

All points in the input file are different and their coordinates are integers not greater than 1000 by the absolute value.

Output

The first line of the output file should contain the single integer K – the number of vertices of the best dog's route. The second line should contain K pairs of coordinates X''_1, Y''_1, ..., X''_K, Y''_K, separated by spaces, that represent this route. If there are several such routes, then you may write any of them.

Sample input

```
4 5
1 4 5 7 5 2 -2 4
-4 -2 3 9 1 2 -1 3 8 -3
```

Example of the output for the sample input

```
6
```

```
1 4 3 9 5 7 5 2 1 2 -2 4
```

Problem E: Guessing Game

Stan and Ollie are playing a guessing game. Stan thinks of a number between 1 and 10 and Ollie guesses what the number might be. After each guess, Stan indicates whether Ollie's guess is too high, too low, or right on.

After playing several rounds, Ollie has become suspicious that Stan cheats; that is, that he changes the number between Ollie's guesses. To prepare his case against Stan, Ollie has recorded a transcript of several games. You are to determine whether or not each transcript proves that Stan is cheating.

Standard input consists of several transcripts. Each transcript consists of a number of paired guesses and responses. A guess is a line containing single integer between 1 and 10, and a response is a line containing "too high", "too low", or "right on". Each game ends with "right on". A line containing 0 follows the last transcript.

For each game, output a line "Stan is dishonest" if Stan's responses are inconsistent with the final guess and response. Otherwise, print "Stan may be honest".

Sample Input

```
10
too high
3
too low
4
too high
2
right on
5
too low
7
too high
6
right on
0
```

Output for Sample Input

```
Stan is dishonest
Stan may be honest
```

G. Cormack

Problem C: Hot Spot

Hot Spot is a single player game played on a 4 by 4 game board. The purpose of the game is to move a red robot from its current location on the board to the top left corner. The game board may also contain green and blue robots. Each square of the game board can be occupied by no more than one robot at any time.

A robot may move in one of two ways:

1. If two robots are adjacent horizontally or vertically (but not diagonally), one of them may jump over the other to the immediately adjacent square, provided that square is empty. For example, if robot a is immediately to the left of robot b, robot a may jump to the square immediately to the right of robot b.
2. If three robots are adjacent in a line (again not diagonally), one of them may jump over the other two, provided the destination square is empty. For example, if robot b is immediately to the right of robot a and robot c is immediately to the right of robot b, robot a may jump to the square immediately to the right of robot c.

Every jump only changes the positions of the existing robots; robots are never removed from or added to the game board.

A blue robot is never allowed to be adjacent horizontally or vertically to another blue robot or to the red robot.

Given the initial configuration of the game board, determine the minimum number of jumps required to move the red robot into the top left corner.

Input Specification

The input begins with a single integer **N**, followed by **N** cases. The input for each specifies the initial position of the board using four lines, each containing four characters. Each character may be either R, indicating the red robot, B, indicating a blue robot, G, indicating a green robot, or a period (.), indicating an empty square.

Sample Input

```
1
.GR.
....
....
....
```

Output Specification

For each case, output a single line containing a single integer, the minimum number of jumps required for the red robot to reach the top left square of the game board.

Output for Sample Input

```
1
```

Ondřej Lhoták

Problem C: Life Forms

You may have wondered why most extraterrestrial life forms resemble humans, differing by superficial traits such as height, colour, wrinkles, ears, eyebrows and the like. A few bear no human resemblance; these typically have geometric or amorphous shapes like cubes, oil slicks or clouds of dust.

The answer is given in the 146th episode of *Star Trek - The Next Generation*, titled *The Chase*. It turns out that in the vast majority of the quadrant's life forms ended up with a large fragment of common DNA.

Given the DNA sequences of several life forms represented as strings of letters, you are to find the longest substring that is shared by more than half of them.

Standard input contains several test cases. Each test case begins with $1 \le n \le 100$, the number of life forms. *n* lines follow; each contains a string of lower case letters representing the DNA sequence of a life form. Each DNA sequence contains at least one and not more than 1000 letters. A line containing 0 follows the last test case.

For each test case, output the longest string or strings shared by more than half of the life forms. If there are many, output all of them in alphabetical order. If there is no solution with at least one letter, output "?". Leave an empty line between test cases.

Sample Input

```
3
abcdefg
bcdefgh
cdefghi
3
xxx
yyy
zzz
0
```

Output for Sample Input

```
bcdefg
cdefgh
```

?

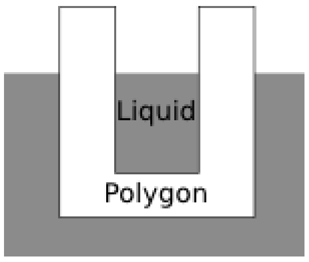
Gordon V. Cormack

Problem D: Meltdown

A polygon is lowered at a constant speed of *v* metres per minute from the air into a liquid that dissolves it at a constant speed of *c* metres per minute from all sides. Given a point *(x,y)* inside the polygon that moves with the polygon, determine when the liquid reaches the point.

The border between air and liquid always has y-coordinate 0, and the liquid only eats away from the sides of the polygon in 2 dimensions. The polygon does not rotate as it is lowered into the liquid, and at time 0, it is not touching the liquid.

Unlike the polygon, which is flat (2-dimensional), the liquid exists in three dimensions. Therefore, the liquid seeps into cavities in the polygon. For example, if the polygon is "cup-shaped", the liquid can get "inside" the cup, as in the diagram below.



Input Specification

The input consists of several test cases.

The first line of each test case contains the five integers *N*, *x*, *y*, *v*, and *c*, where 3 <= *N* <= 30, -100 <= *x* <= 100, 1 <= *y* <= 100, and 1 <= *c* < *v* <= 10.

The following *N* lines of the test case each contain one vertex of the polygon. The i^{th} line contains the two integers *x*, *y*, where -100 <= *x* <= 100, 1 <= *y* <= 100.

The vertices of the polygon are given in counter-clockwise order. The border of the polygon does not intersect or touch itself, and the point *(x,y)* lies strictly inside the polygon—it does not lie on the border of the polygon.

Input is terminated by a line containing `0 0 0 0 0`. These zeros are not a test case and should not be processed.

Sample Input

```
4 0 50 2 1
-1 10
1 10
1 90
-1 90
0 0 0 0 0
```

Output Specification

For each test case, output the first time in minutes that the liquid reaches the specified point, rounded to four decimal places.

Output for Sample Input

```
25.8660
```

Malcolm Sharpe

Problem E - Polylops

Given the vertices of a non-degenerate polygon (no 180-degree angles, zero-length sides, or self-intersection - but not necessarily convex), you must determine how many distinct lines of symmetry exist for that polygon. A line of symmetry is one on which the polygon, when reflected on that line, maps to itself.

Problem Input

Input consists of a description of several polygons.

Each polygon description consists of two lines. The first contains the integer "n" (3<=n<=1000), which gives the number of vertices on the polygon. The second contains "n" pairs of numbers (an x- and a y-value), describing the vertices of the polygon in order. All coordinates are integers from -1000 to 1000.

Input terminates on a polygon with 0 vertices.

Problem Output

For every polygon described, print out a line saying "Polygon #x has y symmetry line(s).", where x is the number of the polygon (starting from 1), and y is the number of distinct symmetry lines on that polygon.

Sample Input

```
4
-1 0 0 2 1 0 0 -1
3
-666 -42 57 -84 19 282
3
-241 -50 307 43 -334 498
0
```

Sample Output

```
Polygon #1 has 1 symmetry line(s).
Polygon #2 has 0 symmetry line(s).
Polygon #3 has 1 symmetry line(s).
```

Problem C: Pick-up sticks

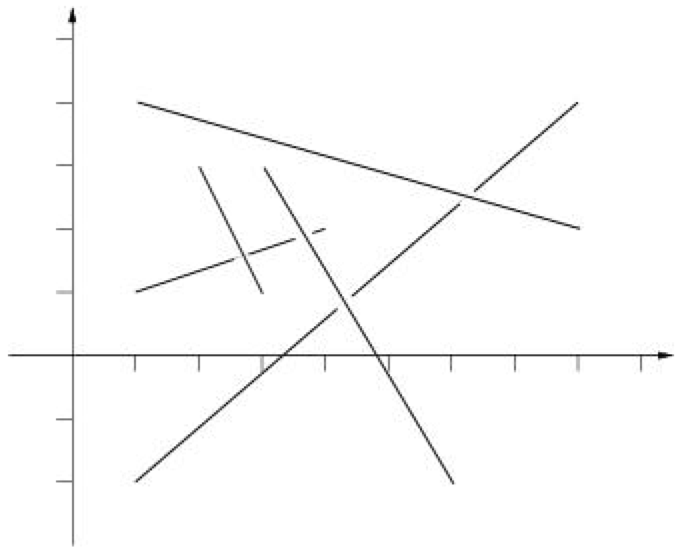
Stan has *n* sticks of various length. He throws them one at a time on the floor in a random way. After finishing throwing, Stan tries to find the top sticks, that is these sticks such that there is no stick on top of them. Stan has noticed that the last thrown stick is always on top but he wants to know all the sticks that are on top. Stan sticks are very, very thin such that their thickness can be neglected.

Input consists of a number of cases. The data for each case start with $1 \leq n \leq 100000$, the number of sticks for this case. The following *n* lines contain four numbers each, these numbers are the planar coordinates of the endpoints of one stick. The sticks are listed in the order in which Stan has thrown them. You may assume that there are no more than 1000 top sticks. The input is ended by the case with $n=0$. This case should not be processed.

For each input case, print one line of output listing the top sticks in the format given in the sample. The top sticks should be listed in order in which they were thrown.

The picture to the right below illustrates the first case from input.

Sample input



```
5
1 1 4 2
2 3 3 1
1 -2.0 8 4
1 4 8 2
3 3 6 -2.0
3
0 0 1 1
1 0 2 1
2 0 3 1
0
```

Output for

sample input

```
Top sticks: 2, 4, 5.
Top sticks: 1, 2, 3.
```

Piotr Rudnicki