Algorithms for Programming Contests
SS20 - Week 12

Chair for Foundations of Software Reliability and Theoretical Computer Science,
TU München

Mikhail Raskin, Tobias Meggendorfer, Stefan Jaax, A.R.Balasubramanian,
Christoph Welzel

Welcome to our practical course! This problem set is due by

Wednesday, 14.07.2020, 6:00 a.m.

Try to solve all the problems and submit them at

`https://judge.in.tum.de/conpra/`

This week's problems are:

The following amount of points will be awarded for solving the problems.

Problem	A	B	C	D	E
Difficulty	easy	easy	medium	medium	hard
Points	4	4	6	6	8

* catchup problem / bonus points, do not count towards total

If the judge does not accept your solution but you are sure you solved it correctly, use the "request clarification" option. In your request, include:

- the name of the problem (by selecting it in the subject field)
- a verbose description of your approach to solve the problem
- the time you submitted the solution we should judge

We will check your submission and award you half the points if there is only a minor flaw in your code.
If you have any questions please ask by using the judge's clarification form.

Problem A
Swordfighting

What a time to be alive. Lea just watched the new "Super Intense Sword Arena" movie. The whole movie was two glorious hours of intense sword dueling between a whole cast of different combatants that would do anything to crush their opponent. At the beginning of each fight, both combatants would be given a random sword from a huge assortment of long swords, broad swords, bastard swords, rapiers, estocs and the like (spoilers: the ones that were lucky enough to get a lightsaber won almost always).

At home, she looks for the frames she found the coolest – every time the two combatants had their swords locked and both tried to push the other one back. After Lea admired the stills for a while, she noticed something – some of the swords did not seem to be straight lines starting from the hilt but rather curved a little. Fascinated she began to investigate and measured several coordinates of the hilts of both swords in the frame. Also she measured the point where both swords clashed (where the sparks fly off).

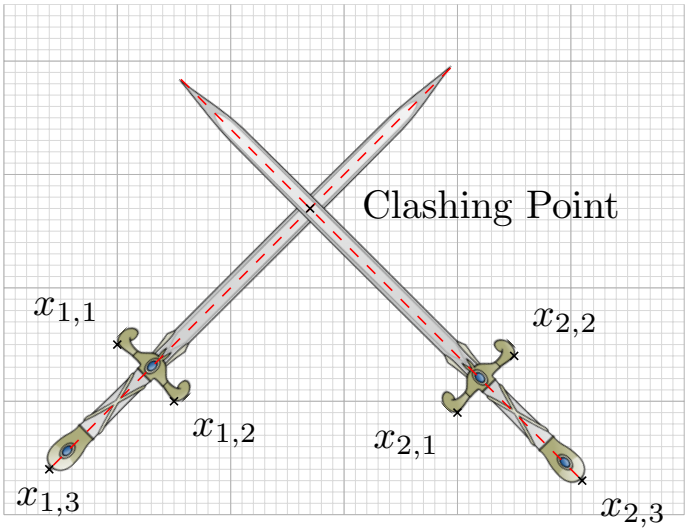


Figure A.1: Visual representation of the first case of the sample input. Sword Image taken from `https://upload.wikimedia.org/wikipedia/commons/4/44/Long_sword.svg`

Can you tell her if the clashing point is consistent – i.e. both swords seem to have straight blades?

Input

The first line of the input contains an integer t. t test cases follow.

Each test case consists of a single line containing 12 integers $x_{1,1}$ $y_{1,1}$ $x_{1,2}$ $y_{1,2}$ $x_{1,3}$ $y_{1,3}$ $x_{2,1}$ $y_{2,1}$ $x_{2,2}$ $y_{2,2}$ $x_{2,3}$ $y_{2,3}$, where the first index specifies which sword it is and the second which part of the sword. Sword parts 1 and 2 are outer points of the crossguard at the same height (i.e. form a line orthogonal to the blade), while 3 is a point at the end of the pommel (see picture). Due to perspective and sword arena physics, you can ignore the width of the blades and think of the blade of the i–th sword as a single straight line extending from the crossguard. $(x_{i,3}, y_{i,3})$ always lies on an imaginary extension of the blade.

Output

For each test case, print a line containing "Case #i: x_c y_c" where i is its number, starting at 1 and (x_c, y_c) is the point where the two blades should clash with an error of up to 10^{-4}. If the blades of both swords do not clash, print a line containing "Case #i: strange". The hilt of a sword never clashes, only the blade. Even if two hilts overlap, it is still "strange". Both swords' blades are long enough to eventually clash. Each line of the output should end with a line

break.

Constraints

- $1 \leq t \leq 20$
- $0 \leq x_{i,j}, y_{i,j} \leq 100$ for all $i \in \{1, 2\}, j \in \{1, 2, 3\}$.
- $y_{i,3} < \min(y_{i,1}, y_{i,2})$ for all $i \in \{1, 2\}$.
- $(x_{i,1}, y_{i,1}) \neq (x_{i,2}, y_{i,2})$ for all $i \in \{1, 2\}$.
- The projection of $(x_{i,3}, y_{i,3})$ onto $(x_{i,1}, y_{i,1}) - (x_{i,2}, y_{i,2})$ lies between $(x_{i,1}, y_{i,1})$ and $(x_{i,2}, y_{i,2})$ for $i \in \{1, 2\}$.
- $(x_{1,3}, y_{1,3}), (x_c, y_c)$ and $(x_{2,3}, y_{2,3})$ will never be collinear.

Sample Input 1

```
2
10 15 15 10 5 4 40 9 45 14 51 3
10 25 20 25 15 10 40 20 50 20 45 15
```

Sample Output 1

```
Case #1: 27.5 26.5
Case #2: strange
```

Sample Input 2

```
8
5 1 1 1 3 0 8 2 10 2 9 0
2 2 4 1 2 0 10 2 7 2 9 1
3 2 5 1 2 0 10 2 7 2 8 1
9 2 6 1 8 0 2 2 4 1 3 0
2 1 4 1 2 0 9 1 7 1 8 0
2 1 4 1 4 0 10 1 8 1 9 0
2 2 4 1 2 0 7 2 10 2 9 0
6 2 9 1 6 0 2 2 5 2 5 1
```

Sample Output 2

```
Case #1: strange
Case #2: 9.0 14.0
Case #3: 8.0 12.0
Case #4: 6.0 6.0
Case #5: strange
Case #6: strange
Case #7: 9.0 14.0
Case #8: strange
```

Problem B
Euler Line

Sometimes, Lea likes to read math books that are written for people without mathematical background and is always fascinated how everything fits together. She recently read a chapter of such a book which was dealing with the so-called Euler line. The Euler line is part of the following theorem shown by Leonhard Euler in 1765: Given any triangle, its orthocenter (the intersection of its altitudes), circumcenter (the intersection of its perpendicular bisectors) and centroid (the intersection of its medians) are colinear, which means they are on one line, the Euler line.

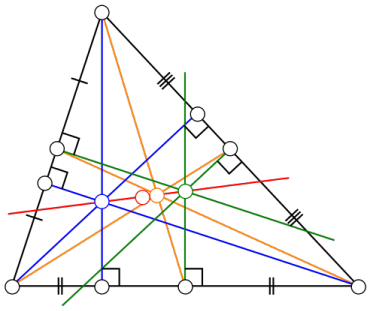


Figure B.1: Euler's line (red) is a straight line through the centroid (orange), orthocenter (blue), circumcenter (green) and even the center of the nine-point circle, which is not considered in this problem (red). Source: wikipedia.org

Lea cannot believe this and thinks this is just a coincidence. The authors of the picture may just have chosen an example where it works. She wants to try this for some other examples and needs you to compute the coordinates of the points in question.

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case consists of three lines, each of them containing two space-separated doubles: The x- and y-coordinates of the triangle's vertices.

Output

For each test case, output one line containing "Case #i:" where i is its number, starting at 1. Afterwards, print the x- and y-coordinates (separated by a space) of the triangle's centroid, orthocenter and circumcenter (in this order), each of them in a seperate line. Each line of the output should end with a line break.

Your solution is considered correct if all coordinates are accurate to six decimal places.

Constraints

- $1 \leq t \leq 20$
- $-100 \leq x, y \leq 100$

Sample Input 1

```
2
0.0 0.0
1.0 0.0
0.0 1.0

1.0 1.0
99.123 -12.5
58.54 -32.643
```

Sample Output 1

```
Case #1:
0.3333333333333333 0.3333333333333333
-0.0 0.0
0.5 0.5
Case #2:
52.88766666666666 -14.714333333333332
49.676013785025596 -97.06973476829143
54.49349310748721 26.463367384145705
```

Sample Input 2

```
11
88.522485 -13.515400
96.557602 -91.472879
84.454638 -4.411959

-3.279037 -78.645062
-52.052942 -29.057519
-15.253739 30.182695

36.046769 -47.450735
26.944309 -66.084199
-61.253595 -93.322007

42.578721 -4.376319
-40.074226 -27.429373
7.812475 -2.407942

-30.804920 -37.877214
43.840443 -47.983403
-62.024012 -46.533710

-48.139515 -63.515330
-64.147930 -97.615190
73.356022 84.524712

-17.834603 84.752420
26.233284 -71.114394
-76.968554 78.692437

-4.323212 68.841193
35.389655 96.911932
43.156818 12.697935

-19.194674 -15.108690
64.847219 -98.297499
-56.392759 -73.380304

93.866003 -83.332342
24.188865 -54.178823
17.522348 -24.369514

-78.935011 31.004570
50.508662 -8.021209
61.379419 24.470746
```

Sample Output 2

```
Case #1:
89.8449083333333 -36.466746000000036
353.4347674713959 23.311959714429097
-41.9500212356979 -66.35609885721455
Case #2:
-23.52857266666667 -25.839962
-78.4333391662827 -31.96024715884536
3.9238105831413415 -22.77981942057733
Case #3:
0.5791609999999942 -68.95231366666665
70.01667024246443 -157.4476219988495
-34.13959362123222 -24.704659500575225
Case #4:
3.4389899999999995 -11.404544666666663
-30.816231489606167 136.08895434024086
20.566600744803086 -85.15129417012045
Case #5:
-16.32949633333335 -44.131442333333325
-27.423852930701237 209.0266633104088
-10.78231803464937 -170.7104951552044
Case #6:
-12.977140999999964 -25.535269333333275
-766.5004672455115 478.8011465743702
363.7845221227557 -277.7034772871851
Case #7:
-22.85662433333333 30.77682099999999
8.407585224917984 102.83064807461625
-38.48872911245898 -5.250092537308133
Case #8:
24.741086999999997 59.483686666666664
2.9952682334270673 69.51618564861288
35.61399638328648 54.46743717569357
Case #9:
-3.580071333333327 -62.262164333333324
-24.561625646796877 -41.222748969507535
6.910705823398435 -72.78187201524624
Case #10:
45.192405333333326 -53.96022633333334
-17.57611725297092 -108.25511995444622
76.57666662648545 -26.812779522776914
Case #11:
10.984356666666663 15.818035666666663
50.312332131187105 -12.237409747748446
-8.679631065593552 29.84575837387423
```

Problem C
Fallingwater

Most likely, you have heard of the famous architectural masterpiece, Fallingwater. Frank Lloyd Wright designed it in 1935 and beautifully integrated the natural flow of water into the house.

Figure C.1: Fallingwater (Kaufmann Residence) by Frank Lloyd Wright.

Lea has planned something much like this for her own house. She will use a waterfall and guide its flow by building stone ledges. She has just one problem: She is unsure where the water will end up. Can you help her?

For this problem, consider the waterfall as two-dimensional. The ledges are given by their start- and endpoint. Water falls down from a given source towards the ground and can never flow upwards. If it touches a ledge it will flow downwards or horizontally along the edge. On horizontal edges, the water splits and continues in both directions. If the point where the water hit the ledge is exactly an endpoint and furthermore the ledge does not go upwards, the water also splits, some of it continues to fall while the rest flows along the edge. See the sketch below.

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case starts with a line containing three integers n x y where n is the number of ledges, and (x, y) is the location of the water source which is considered to be a single point. n lines follow describing the ledges. The i-th line contains four integers $x_{i,1}$ $y_{i,1}$ $x_{i,2}$ $y_{i,2}$ meaning that the i-th ledge goes from $(x_{i,1}, y_{i,1})$ to $(x_{i,2}, y_{i,2})$ in a straight line.

Output

For each test case, print a line containing "Case #i: x" where i is its number, starting at 1, and x is a space-separated, naturally ordered list of the x-coordinates at which water will arrive at the ground ($y = 0$). Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 100$

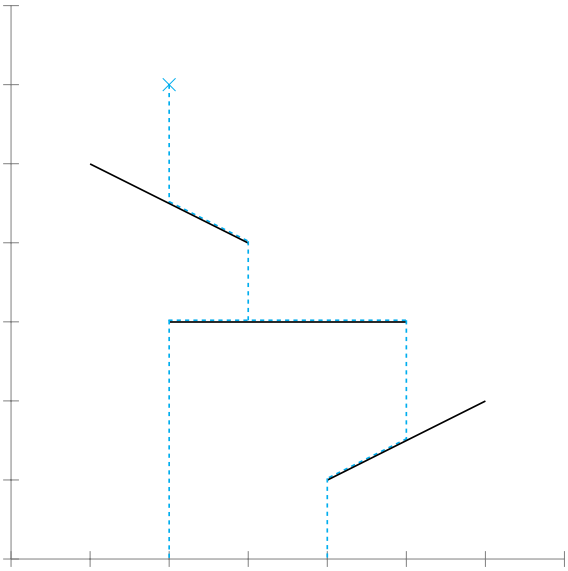


Figure C.2: Case #1

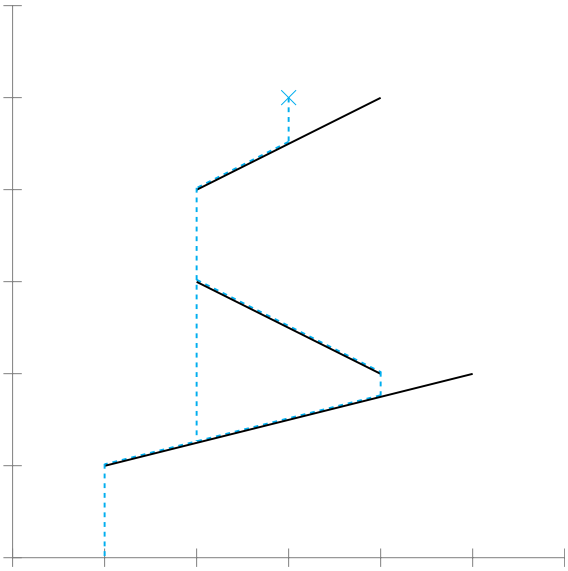


Figure C.3: Case #2

Figure C.4: Illustration of the sample inputs.

- $1 \leq x, y, x_{i,j}, y_{i,j} \leq 100$ for all $1 \leq i \leq n$,$1 \leq j \leq 2$.
- $x_{i,1} \neq x_{i,2}$ for all $1 \leq i \leq n$.
- No two ledges will intersect. Every two points on different ledges are at least 10^{-4} apart.
- The water source is at least 10^{-4} away from each ledge.

Sample Input 1	**Sample Output 1**
2 3 2 6 1 5 3 4 2 3 5 3 4 1 6 2 3 3 5 2 4 4 5 2 3 4 2 1 1 5 2	Case #1: 2 4 Case #2: 1

Sample Input 2

```
5
5 8 10
3 1 6 1
4 5 7 1
7 2 8 1
3 3 5 2
2 5 6 2

5 4 10
3 5 10 1
3 1 6 1
8 2 9 1
9 3 10 2
5 2 6 2

3 5 10
4 4 9 1
5 1 6 2
3 1 4 1

3 3 10
9 1 10 4
3 4 4 1
6 5 7 1

5 3 10
2 4 10 1
1 3 7 1
8 4 9 2
6 6 10 3
2 7 5 3
```

Sample Output 2

```
Case #1: 8
Case #2: 10
Case #3: 9
Case #4: 3 4
Case #5: 10
```

This page is intentionally left blank.

Problem D
Family Pictures

While visiting her childhood home, Lea found a box of old family pictures in the attic. Looking at some pictures, she notices that she seemed to grow up quite quickly. But how quickly exactly? For a thorough scientific analysis, Lea therefore wants wants to measure her body height by looking at the pictures from different years.

Unfortunately, many of the old pictures were still taken using reversal film and are stored on diapositives, small slides the size of about $5 \times 5\,\text{cm}^2$. It's hard to measure her size from such a small image. So Lea took her family's old slide projector and projected the images on a large wall.

Scaling the pictures up worked well — however the lens of the projector, which hasn't been used in decades, had several defects. The projected images are quite distorted: they are skewed, shifted, tilted, or even rotated! Especially, ratios of length and parallel lines are not preserved in general. Of course, this makes it hard to measure lengths. However, Lea notices that at least, lines are preserved, i.e. any three points that were on the same line in the image are also on the same line in the projection.

Lea is sure that by having the coordinates of certain points on the wall, she can recover the length information in the original image. She proceeds to measure the four corner points of the originally square image on the wall. Additionally, she measures the point at the top of her head, and the center point of a clock on the wall she was standing next to, which she knows was always at the height of exactly $1\,\text{m}$.

Given these six points, can you tell Lea how large she is in the picture?

Input

The first line of the input contains an integer t. t test cases follow.

Each test case consists of a single line containing twelve decimal numbers $a_x\ a_y\ b_x\ b_y\ c_x\ c_y\ d_x\ d_y\ e_x\ e_y\ f_x\ f_y$, where

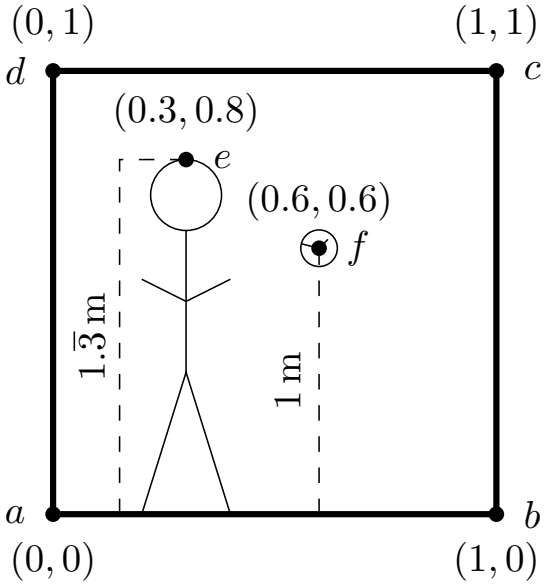
- the point (a_x, a_y) is the projection of the lower-left corner at $(0, 0)$ in the original image,
- the point (b_x, b_y) is the projection of the lower-right corner at $(1, 0)$ in the original image,
- the point (c_x, c_y) is the projection of the upper-right corner at $(1, 1)$ in the original image,
- the point (d_x, d_y) is the projection of the upper-left corner at $(0, 1)$ in the original image,
- the point (e_x, e_y) is the projection of the point at the top of Leas head,
- the point (f_x, f_y) is the projection of the center of the wall clock at the height of $1\,\text{m}$.

Output

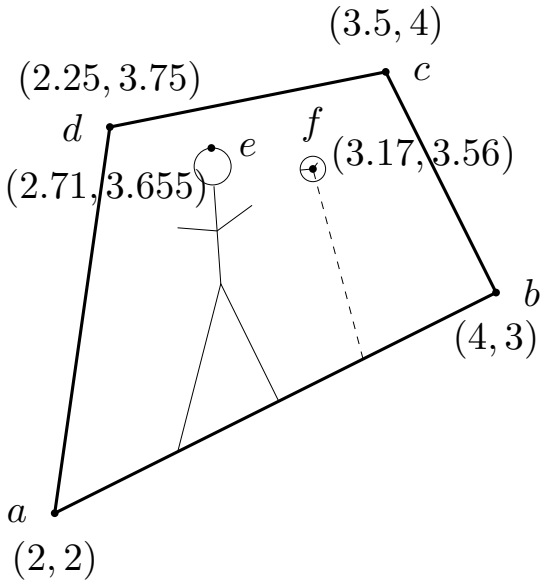
For each test case, output one line containing "Case #i: x" where i is its number, starting at 1, and x is the height of Lea in the original picture frame in meters, with an absolute error of up to 10^{-4}.

Constraints

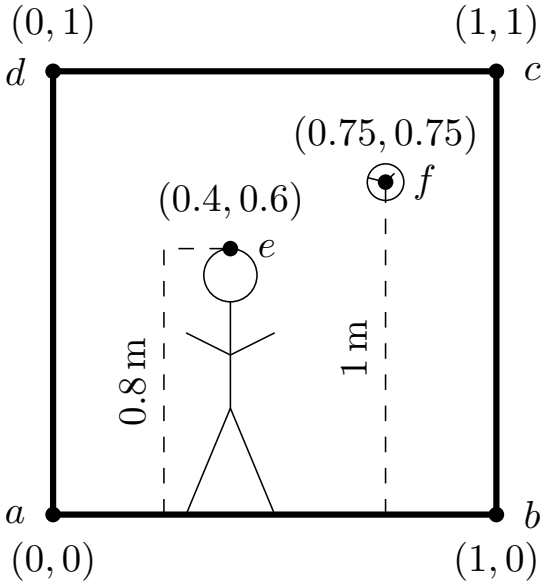
- $1 \leq t \leq 20$
- $0.0 \leq a_x, a_y, b_x, b_y, c_x, c_y, d_x, d_y, e_x, e_y, f_x, f_y \leq 10.0$
- No three of the four points $(a_x, a_y), (b_x, b_y), (c_x, c_y), (d_x, d_y)$ are collinear.
- The quadrilateral $(a_x, a_y), (b_x, b_y), (c_x, c_y), (d_x, d_y)$ is convex.
- The points (e_x, e_y) and (f_x, f_y) are strictly on the inside of the quadrilateral $(a_x, a_y), (b_x, b_y), (c_x, c_y), (d_x, d_y)$.



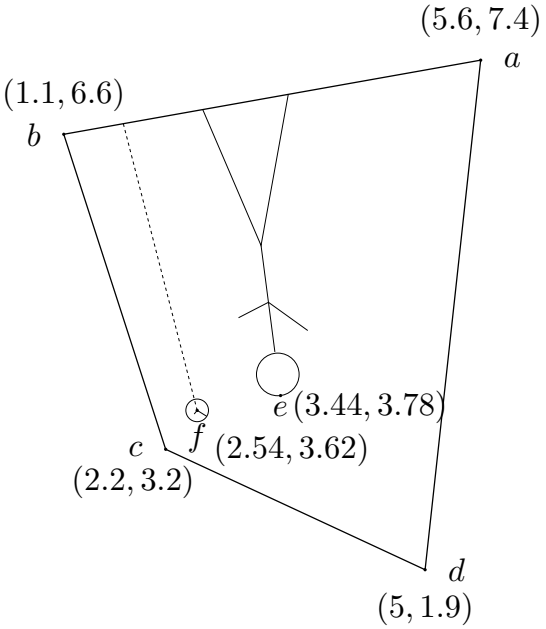
Case #1: Original image.



Case #1: Projected image.



Case #2: Original image.



Case #2: Projected image.

Figure D.1: Illustration of the first sample input. The input are the coordinates in the projected image on the right. The output is the height of Lea in the original image on the left.

Sample Input 1	**Sample Output 1**
2 2 2 4 3 3.5 4 2.25 3.75 2.71 3.655 3.17 3.56 5.6 7.4 1.1 6.6 2.2 3.2 5 1.9 3.44 3.78 2.54 3.62	Case #1: 1.333333333 Case #2: 0.8

Sample Input 2

```
10
3.77 8.02 1.35 5.02 6.37 4.14 5.70 6.38 4.55 6.23 3.75 6.83
7.70 6.97 3.23 6.52 2.31 2.51 7.63 2.10 5.34 5.34 3.48 3.52
2.65 8.48 3.46 4.91 9.28 7.79 9.17 9.95 5.67 7.28 9.18 7.96
1.52 4.46 2.54 1.77 8.22 6.64 0.62 9.15 2.02 7.90 4.51 4.03
5.48 4.46 7.32 7.53 1.77 7.58 1.85 0.49 6.16 5.71 5.27 5.92
8.41 3.86 8.23 6.78 0.35 2.06 6.34 2.69 5.51 5.15 5.26 4.36
8.50 6.35 2.48 8.13 1.16 3.98 2.24 0.91 5.78 5.53 4.04 5.67
0.34 6.15 9.26 7.22 9.21 9.76 3.78 8.18 9.06 7.81 1.06 6.45
4.73 9.87 5.02 2.72 9.61 3.74 9.34 9.60 6.46 7.08 7.82 7.59
7.70 1.01 7.64 5.59 2.28 5.72 5.66 1.88 6.45 3.38 6.94 4.20
```

Sample Output 2

```
Case #1: 2.717823095
Case #2: 0.466630744
Case #3: 0.2961857833
Case #4: 1.070240563
Case #5: 0.142460224
Case #6: 0.7522650777
Case #7: 0.5915934551
Case #8: 1.368807465
Case #9: 0.5080808653
Case #10: 2.288161243
```

This page is intentionally left blank.

Problem E
Fractals

Since Lea likes to draw and she also likes mathematics, there is nothing more obvious than the fact that Lea absolutly loves fractals. Her newest project is a huge fractal in her garden: She wants to walk around the lawn and her footsteps should form a nice fractal that can be seen from her roof terrace and even from outer space!

The problem is that she needs clear instructions on how to walk since it is hard to see the fractal's structure if you stand inside. Lea found a solution for this problem. This is how she creates her walking instructions: She starts by choosing an integer d, some word s, an angle a and a set of n productions p. A production is a mapping from a letter to a word containing other letters or the characters "+" or "-". Now, she replaces all letters in s by the strings given by the corresponding productions, "+" and "-" are not replaced. She repeats this process d times.

Afterwards, she starts walking using the string she just computed. She reads the characters one after another from left to right. If she reads a "+" sign, she turns a degress to the left. If she reads a "-" sign, she turns a degress to the right. If she reads some other character, she walks 1 meter in her current direction.

Lea wants to see the result of her inputs first without walking long distances. Can you tell her how the resulting structure will look like?

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case starts with a single line containing space-separated integers n, d, a and the string s. n lines follow describing the productions. Each line has the form $x => y$ where x is a character and y is a string meaning that x is replaced by y.

Output

For each test case, print a line containing "Case #i:" where i is its number, starting at 1. Afterwards, print each point Lea will move to starting at $(0, 0)$ in a single line. Lea starts moving into the direction of point $(1, 0)$. Each point should be given as a space-separated list of its coordinates. Each line of the output should end with a line break. Your answer will be considered correct if the absolute error of each number in the output is at most 10^{-2}.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 26$
- $1 \leq d \leq 15$
- $1 \leq a \leq 359$
- All strings of the input consist of 1 to 20 upper case letters, "+" or "-".
- There is a production for every letter that appears in the input.
- The output of each test case contains at most 10^6 points.

These picures were generated with GNUplot. You can create the first picture with the following command:

```
gnuplot -e "plot 'data.out' every 1::1::9 \
        title 'sample.out, Case \#1' with lines; pause -1"
```

Adjust the line numbers to draw the other test cases. Here are some other examples not mentioned in the sample input:

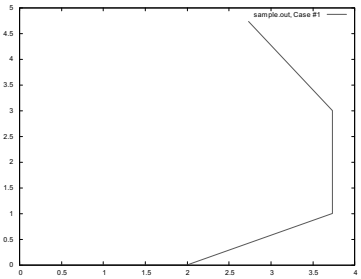


Figure E.1: Case #1

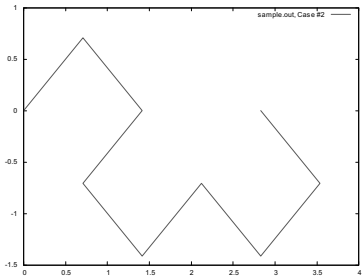


Figure E.2: Case #2

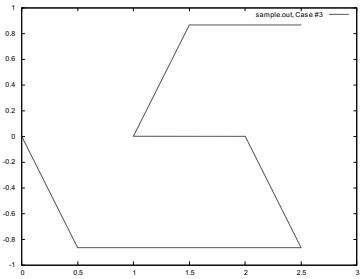


Figure E.3: Case #3

Figure E.4: Illustration of the sample inputs.

Sample Input 1

```
3
1 3 30 L
L=>LL+

2 3 45 L
R=>+R--L+
L=>-R++L-

2 1 60 L
R=>R+L++L-R--RR-L+
L=>-R+LL++L+R--R-L
```

Sample Output 1

```
Case #1:
0.0 0.0
1.0 0.0
2.0 0.0
2.8660254037844393 0.5
3.7320508075688776 1.0
3.732050807568879 2.0
3.7320508075688785 3.0
3.2320508075688785 3.8660254037844384
2.7320508075688785 4.732050807568877
Case #2:
0.0 0.0
0.7071067811865476 0.7071067811865475
1.4142135623730951 0.0
0.7071067811865479 -0.7071067811865478
1.4142135623730954 -1.4142135623730951
2.121320343559643 -0.7071067811865474
2.8284271247461903 -1.414213562373095
3.535533905932738 -0.7071067811865472
2.8284271247461916 8.881784197001252E-16
Case #3:
0.0 0.0
0.5000000000000001 -0.8660254037844386
1.5 -0.8660254037844386
2.5 -0.8660254037844386
2.0000000000000004 -1.1102230246251565E-16
1.0000000000000004 3.3306690738754696E-16
1.5000000000000007 0.8660254037844389
2.500000000000001 0.8660254037844389
```

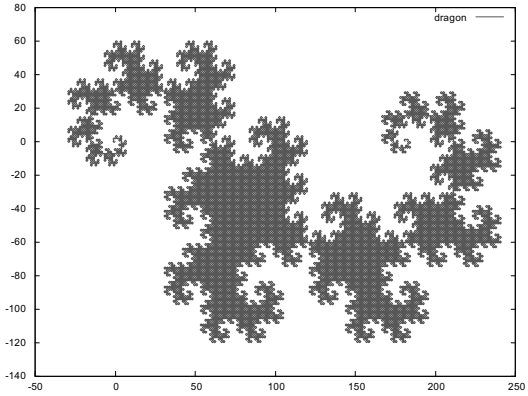


Figure E.5: Dragon

```
1
2 15 45 L
R=>+R--L+
L=>-R++L-
```

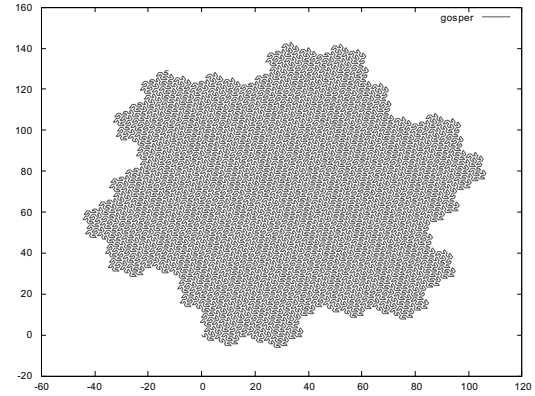


Figure E.6: Gosper

```
1
2 5 60 L
R=>R+L++L-R--RR-L+
L=>-R+LL++L+R--R-L
```

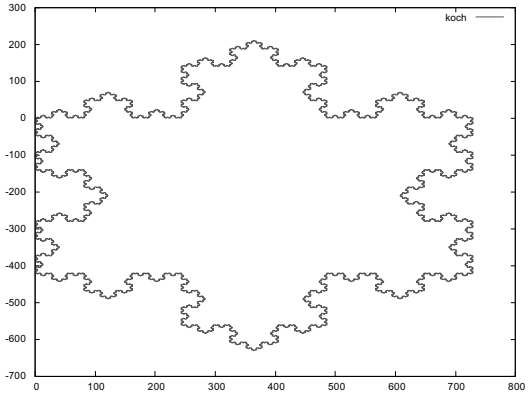


Figure E.7: Koch

```
1
1 6 60 F--F--F
F=>F+F--F+F
```

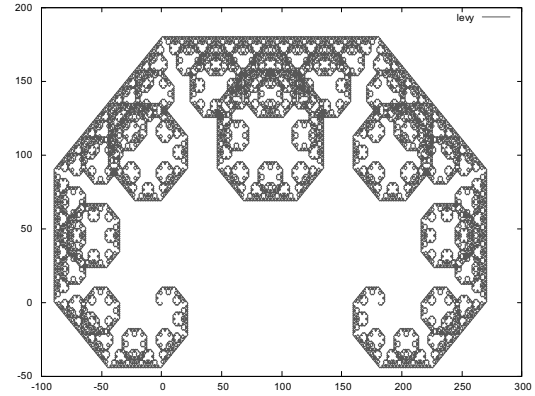


Figure E.8: Levy

```
1
1 15 45 F
F=>+F--F+
```

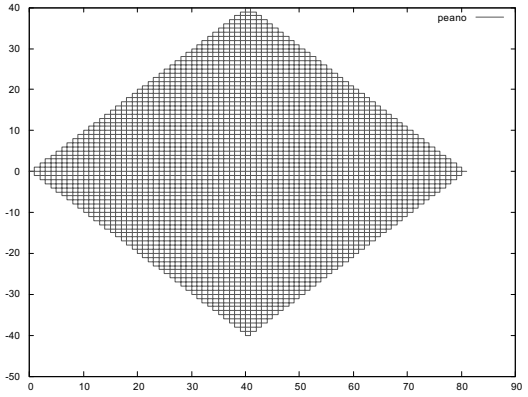


Figure E.9: Peano

```
1
1 4 90 F
F=>F-F+F+F+F-F-F-F+F
```

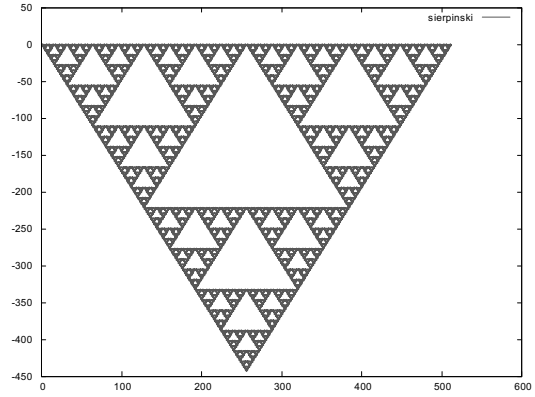


Figure E.10: Sierpinski

```
1
2 8 60 FXF--FF--FF
X=>--FXF++FXF++FXF--
F=>FF
```

Figure E.11: Illustration of additional inputs.