Algorithms for Programming Contests
WS15/16 - Week 13

Chair for Efficient Algorithms (LEA), TU München

Prof. Dr. Harald Räcke

Moritz Fuchs, Philipp Hoffmann, Christian Müller, Chris Pinkau, Stefan Toman

This problem set is due by

Wednesday, 27.01.2016, 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

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.

This page is intentionally left blank.

Problem A
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.

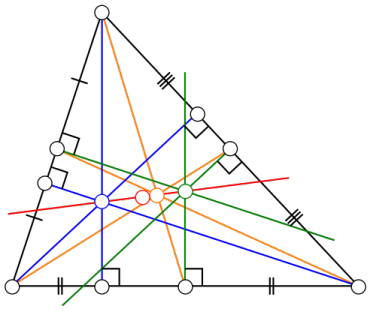

Figure A.1: Euler's line (red) is a straight line through the centroid (orange), orthocenter (blue), circumcenter (green) and even the center of the nine-point circle, which is not considered in this problem (red). Source: wikipedia.org

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 B
Templonian Excavation

On her visit to Templonia, Lea comes by an old Templonian temple site, which has not yet been excavated completely. Fascinated, she spends the day talking to the archaeologists that are busily scouring the area, unearthing artifacts from ancient times.

In those times in Templonia, the number 3 was a very holy number. That is why they raised many monuments to it. The excavation site Lea visited is one of them.

It consists of several altars, each of them surrounded by 3 obelisks that form a triangle. Inside that triangle, they paved the floor with triangular floor tiles and for every 3 square meters of area of the triangle they set up a 3-armed candelabra (Since the area generally was not divisible by 3, they rounded the number of square meters to the nearest multiple of 3 and since they could never have enough 3s they also rounded the resulting number of candelabras to the nearest multiple of 3). They then held mass inside these triangles where they chanted songs about the holy 3 in which 3 viciously slaughtered every other number until it was the only number left. Then, they sacrificed 3 virgins, drank 3 different kinds of religious wine and danced until morning. All in all it sounded like a big party.

While she heard about all this, she watched the archaeologists uncover the floor tiles, candelabras, goblets, etc. As she really liked the old candelabras she wanted to know how many more the archaeologists would probably find.

Input

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

Each test case consists of 6 integers: $x_1\ y_1\ x_2\ y_2\ x_3\ y_3$. The points $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ specify the locations of the 3 obelisks (coordinates are given in meters).

Output

For each test case, output one line containing "Case #i: $number$" where i is its number, starting at 1, and $number$ is the amount of candelabras the Templonians used to illuminate their sacrificial parties. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $0 \leq x_i, y_j \leq 10000$
- $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ are pairwise distinct
- $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ form a convex area

Sample Input 1	Sample Output 1
4 0 0 3 0 0 3 0 0 6 0 0 6 0 0 5 0 0 4 15 16 17 14 30 40	Case #1: 3 Case #2: 6 Case #3: 3 Case #4: 12

Sample Input 2	Sample Output 2
10 1 10 8 10 6 1 1 4 10 5 7 6 1 9 10 7 2 10 2 4 6 6 7 7 6 1 6 5 10 6 2 10 10 0 8 5 3 0 7 8 3 8 6 2 8 2 3 4 7 3 10 8 2 6 6 5 9 4 6 4	Case #1: 12 Case #2: 3 Case #3: 3 Case #4: 0 Case #5: 3 Case #6: 3 Case #7: 6 Case #8: 0 Case #9: 6 Case #10: 0

Problem C
Area 51

Lea saw an UFO yesterday night. You do not believe her? She is absolutely convinced that these red and blue flashing lights she saw were made by aliens. Now, she wants to find out more about this UFO, but where to find it? She had been thinking about this problems for some hours and finally found the answer: Area 51. If the UFO crashed it is now at Area 51 for sure, that secret military base somewhere in the desert. Nobody knows too much about this military base, so Lea decides to infiltrate it.

At first, she needs to know more about Area 51. How big is it, where are the weak points of the border control, what about the condition of the soil and so on. Her first step is to walk along the whole fence around the building. She logs her GPS coordinates, makes photos and takes notes about the environment.

Later, in her hotel, she wants to evaluate the new data. In particular, she wants to know the size of the area the military base is built upon. Can you compute it for her?

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 a single integer n, the number of GPS points she recorded in order. n lines follow describing the points. The i-th line contains two doubles x_i and y_i, the x and y coordinates of the point, respectively. After the n-th point she got back to her starting position. Between the points, Lea always walked straight lines.

Output

For each test case, output one line containing "Case #i: x" where i is its number, starting at 1, and x is the area of the military base in square kilometers. Each line of the output should end with a line break.

You may assume that the ground is totally flat and that Lea never walked the same place twice, except for the start and end of her trip, in particular, none of the straight lines she walked intersect except at the vertices. One unit in the coordinates is 1km long. Note that there are no further restrictions, in particular, the area may be highly non-convex.

Your solution is considered correct if the area is accurate to four decimal places.

Constraints

- $1 \leq t \leq 20$
- $3 \leq n \leq 50$
- $-100 \leq x_i, y_i \leq 100$ for all $1 \leq i \leq n$

Sample Input 1

```
3
3
0.0 0.0
1.0 0.0
0.0 1.0

3
0.0 0.0
0.0 1.0
1.0 0.0

5
0.0 0.0
0.0 2.0
1.0 1.0
2.0 2.0
2.0 0.0
```

Sample Output 1

```
Case #1: 0.5
Case #2: 0.5
Case #3: 3.0
```

Sample Input 2

```
4
4
-58.07466328205631 9.405672395543618
76.6319488338469 15.108858541202835
-53.443952147392 72.45288779108455
-78.93541967722052 68.32425998564514

6
70.54369550409413 -79.53062499560784
-56.37704753822208 -20.583817498762485
-87.37193555855474 -91.6436211495207
-67.54667759854604 74.29745725658617
81.93690436983397 -29.181281743280124
30.663829854406885 -13.413894994507714

6
-20.104365045221485 11.470846571974562
-7.140443639951741 -74.62149905630542
83.02974590501162 -78.42280813534694
-17.8169258450136 76.23171334405177
-88.3636586515233 21.426358677652118
16.118849162254207 9.278147106838446

3
-5.843922497460198 -45.87423560320103
-9.695281188689037 -9.830085438987652
-79.53335898306378 -5.952743811648901
```

Sample Output 2

```
Case #1: 5027.257271519586
Case #2: 10531.341185904002
Case #3: 9415.359528470672
Case #4: 1251.1605649125722
```

Problem D
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 D.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

- $1 \leq t \leq 20$

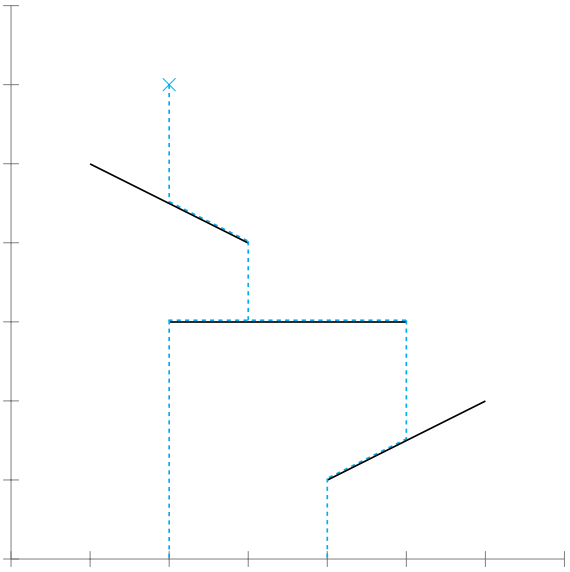

Figure D.2: Case #1

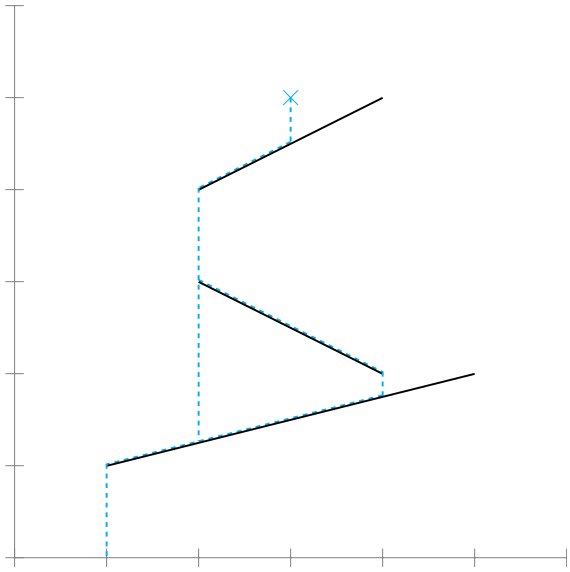

Figure D.3: Case #2

Figure D.4: Illustration of the sample inputs.

- $1 \leq n \leq 100$
- $1 \leq x, y, x_{i,j}, y_{i,j} \leq 100$ 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

```
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
```

Sample Output 1

```
Case #1: 2 4
Case #2: 1
```

Sample Input 2	Sample Output 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	Case #1: 8 Case #2: 10 Case #3: 9 Case #4: 3 4 Case #5: 10

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:

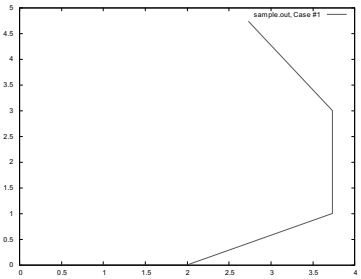

Figure E.1: Case #1

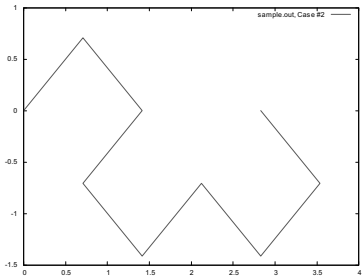

Figure E.2: Case #2

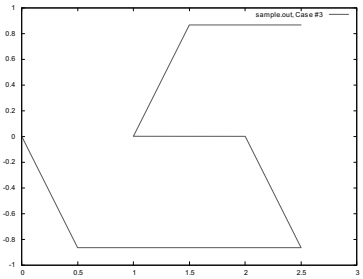

Figure E.3: Case #3

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
```

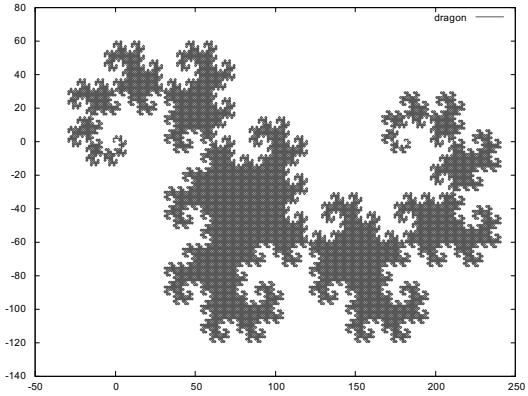

Figure E.5: *
```
1
2 15 45 L
R=>+R-L+
L=>-R++L-
```

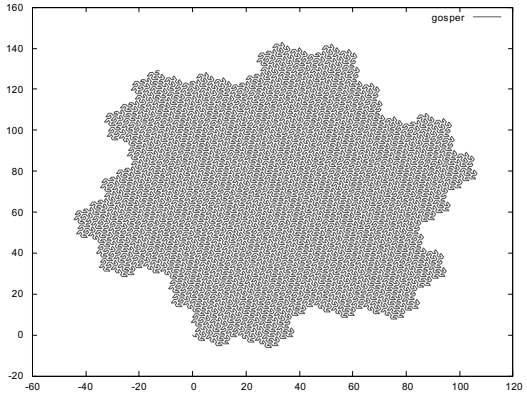

Figure E.6: *
```
1
2 5 60 L
R=>R+L++L-R-RR-L+
L=>-R+LL++L+R-R-L
```

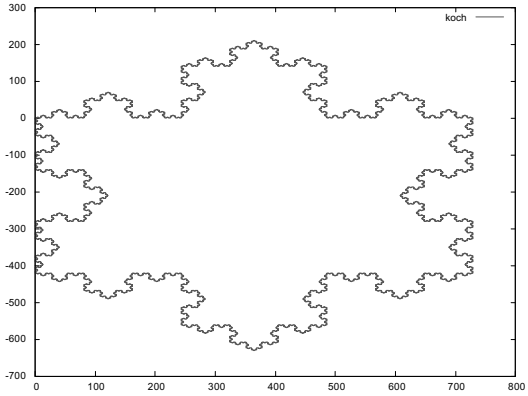

Figure E.7: *
```
1
1 6 60 F-F-F
F=>F+F-F+F
```

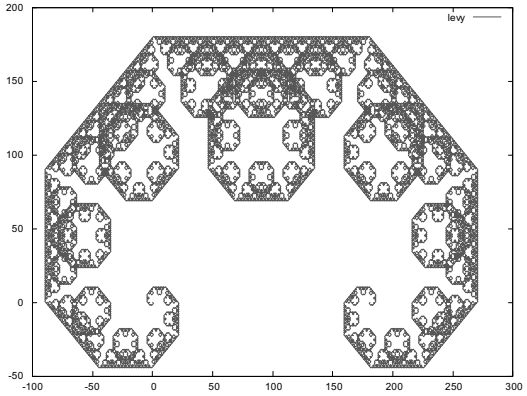

Figure E.8: *
```
1
1 15 45 F
F=>+F-F+
```

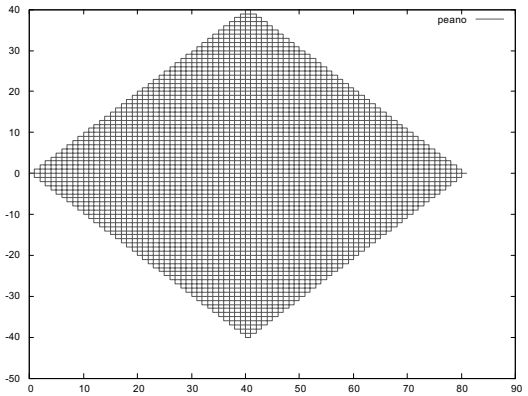

Figure E.9: *
```
1
1 4 90 F
F=>F-F+F+F+F-F-F-F+F
```

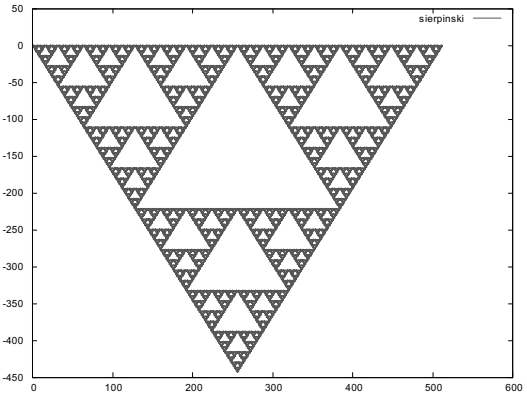

Figure E.10: *
```
1
2 8 60 FXF-FF-FF
X=>-FXF++FXF++FXF-
F=>FF
```

Figure E.11: Illustration of additional inputs.