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:

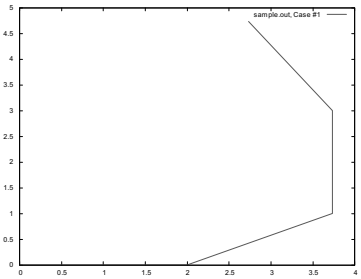

Figure 1: Case #1

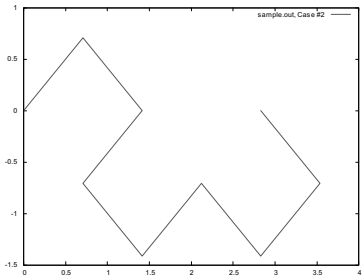

Figure 2: Case #2

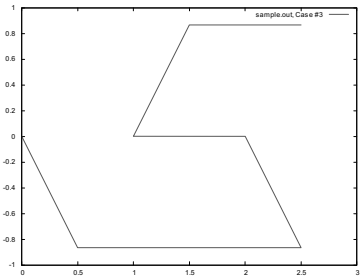

Figure 3: Case #3

Figure 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
```

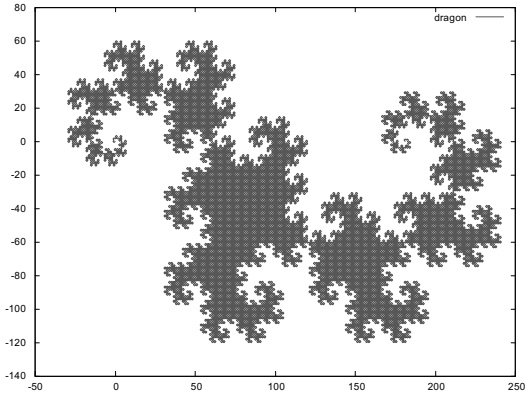

Figure 5: Dragon

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

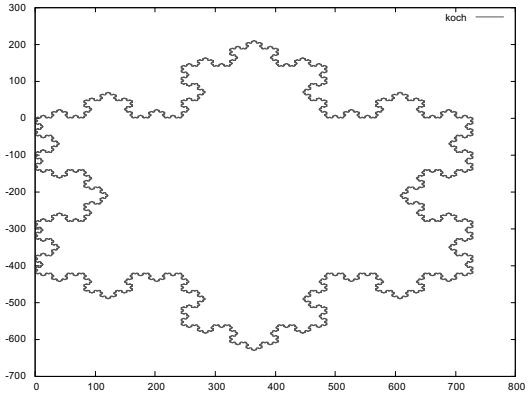

Figure 7: Koch

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

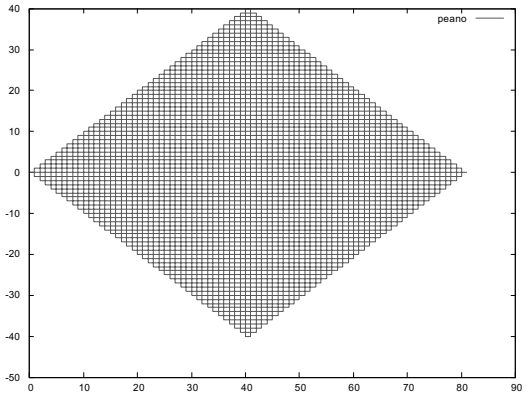

Figure 9: Peano

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

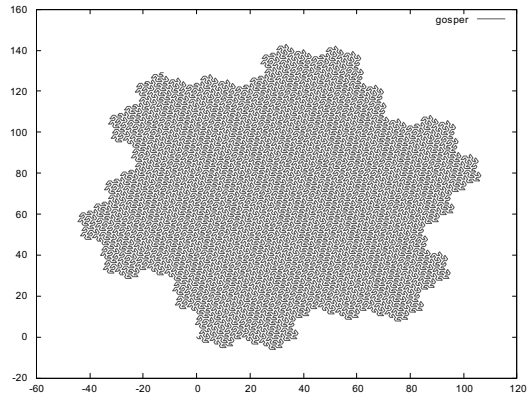

Figure 6: Gosper

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

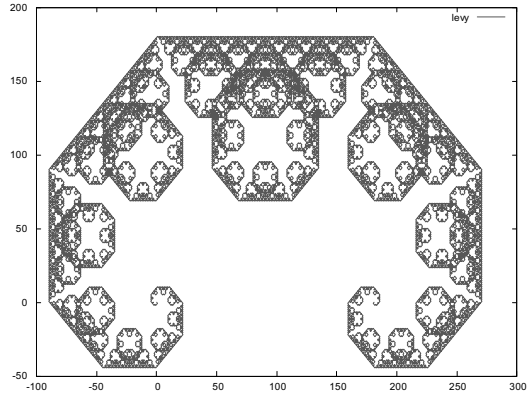

Figure 8: Levy

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

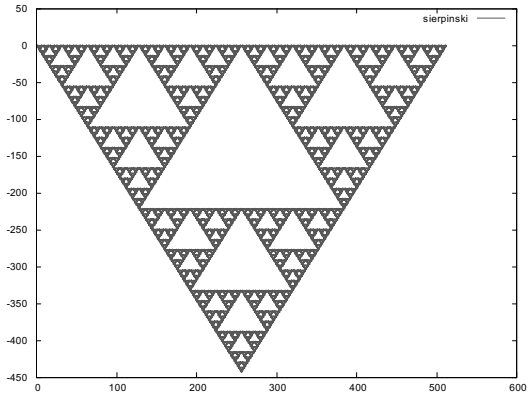

Figure 10: Sierpinski

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

Figure 11: Illustration of additional inputs.