Ants on a plane!

In this étude you are going to be simulating (some of) the behaviour of creatures related to Langton's ant. Imagine an (in principle) infinite plane of square tiles, with a lonely ant initially standing at $(0, 0)$. The ant will take a certain sequence of steps of unit length in one of the four compass directions determined by various rules (*i.e.*, its "DNA"). These rules specify the direction of the next step based on the previous step, and the state of the ant's current position—they may also specify a change in that state.

For instance, Langton's original Ant lives on a plane with two possible states (black and white) and with the rules "right on white, left on black, flip the colour".

The problem you have to solve here is, given an ant's DNA, determine its location after a certain number of steps (starting on a plane whose points are all of some given state).

Task

In this task, input will come from `stdin` and will consist of a sequence of "scenarios" separated from one another by empty lines. There may also be some lines beginning with the hash character, #. These are comments and should be ignored entirely. A comment does *not* count as an empty line. The output for each scenario, specified below, should be written to `stdout`.

An individual scenario consists of the DNA of an ant, followed by a positive integer representing the number of steps you are to follow the ant for. For Langton's ant, the DNA is represented as follows:

```
w ESWN bbbb
b WNES wwww
```

Each line of DNA consists of three parts separated by single spaces:

- A single character representing a state.
- A sequence of four compass directions representing the direction of the next step from a point in this state, after arriving with a North, East, South, and West step respectively (so if a Langton's ant arrives at a white point from the south, *i.e.*, using a North step, its next step is East—a right turn).
- A sequence of four characters representing the state of the point after the ant leaves (again based on the incoming direction).

At each step, the ant:

- determines the state of the tile it is standing on;
- finds the rule in its DNA for that state;
- uses its direction to select the new tile state;
- uses its direction to determine its new direction; and
- moves one step in that new direction.

Initially, all points of the plane are presumed to be in the state coded by the first line of the DNA (*e.g.*, `w` above), and the ant is facing North on the point $(0, 0)$ (*i.e.*, its first step is to be determined as if it had arrived at $(0, 0)$ from the south). The "mathematical" coordinate system is used—a North step increases the second (or y) coordinate by 1, and an East step increases the first (or x) coordinate by 1.

It is *not* to be assumed that there will always be just two states, or that they are coded by letters. State names can be *any* non-whitespace character other than "#". For instance this is a perfectly valid DNA representation:

```
. EEEE xx!x
x SEWN !.9x
! WEEE 9999
9 NWES .x!9
```

You *may* assume that the DNA is always complete (i.e., you will not be given next states in the third part that are not represented in the list).

The output for a scenario is to echo the input (except comments), followed by a line consisting of a # character, followed by a space, then the x-coordinate of the ant's final position, another space, and the y-coordinate of the ant's final position. The output of each distinct scenario should be separated from the next by a single empty line.

Standards

For an achieved standard the program must operate as specified for up to 10,000 steps.

Merit criteria include well-structured and readable code, and efficient implementations that allow tracking ants for up to 50,000,000 steps.

Excellence criteria include algorithms that (in some cases) would allow tracking for an essentially arbitrary number of steps, or extensions to the program.

Objectives

1.2, 1.3, 1.4, 2.2, 2.3, 2.6, 2.7, 3.3, 3.4, 3.5, 3.6, 4.1, 4.3.

(Pair)