Starfleet Mine Clearing Exercise Evaluator

Starfleet Academy requires that all cadets take a course on programming robotic mine clearing vessels. You have been selected to teach this course.

The mine clearing vessels are very simple vehicles that move forward at a constant rate, can fire volleys of photon torpedoes, and can make some very simple navigational changes.

When Starfleet officers talk about the space to be cleared of mines by a vessel, they refer to the space as a cuboid, with a given height, width and depth. They talk about how a vessel “falls” through the cuboid at a rate of 1km/s. Each second, the ship can optionally fire a volley of torpedoes in one of four patterns (alpha, beta, gamma, or delta), and can optionally move 1km north, south east or west on the planar cross-section of the cuboid on which it is currently located. “Dropping” through the space is considered equivalent to decrementing the z-coordinate of the ship. The moves on the current planar cross-section, and their meanings, are as follows:

north	increment y-coordinate of ship
south	decrement y-coordinate of ship
east	increment x-coordinate of ship
west	decrement x-coordinate of ship

Each firing pattern is defined by the offsets from the ship's current position at which the torpedoes travel (and can destroy mines). The patterns, and their associated offsets from the ship's current position, are defined as follows:

alpha	(-1, -1), (-1, 1), (1, -1), (1, 1)
beta	(-1, 0), (0, -1), (0, 1), (1, 0)
gamma	(-1, 0), (0, 0), (1, 0)
delta	(0, -1), (0, 0), (0, 1)

Cadets at the academy are given an initial layout of the cuboid of space they are to clear and submit a “script” of firing patterns and moves. You, as their instructor, are to give them feedback on how well their scripts clear the space of mines.

You have decided to write a program that evaluates students' scripts. This program should accept the names of two files on the command line: the name of a “field” file; and the name of a “script” file. The field file describes the cuboid to be cleared. The script file contains actions to be performed at each second. Your program should read these files, run a simulation with the given script, and determine if the script successfully clears the space.

Input

The field file describes the cuboid as a grid. The starting point of the vessel is assumed to be the middle of the grid. The grid consists of characters that represent either empty space or a mine at a particular distance. An `a` character represents a mine that is 1km away; `b` is a mine 2km away, through `z` for a mine 26km away. An `A` character represents a mine 27km away, through `Z` for a mine 52km away. A `.` (period) represents empty space all the way through the cuboid. At most 1 mine can occupy a given coordinate in the cuboid. If two mines share x- and y-coordinates, but have different z-coordinates, only the closest of those mines is represented in the grid. For example,

```
.e.
..a
A..
```

represents a cuboid of 3 x 3 x 27km. Assuming the top-left corner of the grid is coordinate (0, 0, 0), the ship is at (1, 1, 0), and there are mines at (2, 1, -1), (1, 0, -5) and (0, 2, -27).

The script file consists of lines, in which each line has an optional firing pattern and optional move, separated by arbitrary whitespace. After the commands on a line are executed, the ship should "drop" down into the cuboid 1km. For example, the line:

```
alpha
```

means the vessel should fire a volley in the alpha pattern and then "drop" straight down 1km. The line:

```
north
```

means the vessel should move north 1km and then drop down 1km. The line:

```
alpha north
```

means the vessel should fire a volley in the alpha pattern, move north 1km and then drop down 1km. Finally, a blank line (or one that consists only of whitespace) means the vessel should simply drop 1km without firing or moving.

Output

Your evaluation program will run the given script against the given field. For each step in the simulation, your program should print: 1) the string `Step <x>`, where `<x>` is the number of the step being made, with counting starting at 1; 2) the current minefield; 3) the current instruction(s) (i.e., the optional firing pattern and the optional move); and, 4) the resultant field. A single blank line should be printed between each of these and between steps. The simulation is over when: a) all mines are cleared (whether or not there are still instructions to be run); b) the script is completed; or, c) the vessel "passed" a mine (i.e., the z-coordinate of a mine and the vessel match).

Your program will score the given script as well. The score is calculated as follows:

1) passed a mine – fail (0 points)

2) script completed but mines still remaining – fail (0 points)

3) all mines cleared, but steps remaining – pass (1 point)

4) all mines cleared, no steps remaining – pass (n points – see below)

The score for the fourth case is calculated as a function of the number of initial mines, the number of volleys fired, and the number of moves made. The starting score is 10 times the initial number of mines in the cuboid. Subtract 5 points for every shot fired, but no more than 5 times the number of initial mines. Also, subtract 2 points for every km moved north, south east or west (not for kms dropped), but no more that 3 times the number of initial mines. At the end of the simulation, print the result and the score. The last line your evaluation program should print is `<result> (<n>)`, where `<result>` is either `pass` or `fail`, and `n` is the score, calculated as above.

Miscellaneous

When a volley is fired, it destroys every mine with an offset relative to the current position of the vessel effectively instantaneously. A torpedo travels the depth of the cuboid, so even if several mines were to share x- and y-coordinates, the same torpedo would eliminate all of them. There is no “splash” damage from a torpedo. An explosion only destroys mines at the offsets given. A vessel has, for all intents and purposes, enough torpedoes and fuel to fire and move every turn.

When the grid is printed (before and after every move), it should be drawn so that the vessel is in the middle of the grid, and the grid should be no larger than necessary. For example, with the vessel at (0, 0, 0) and mines at (-2, 0, -10) and (1, 1, -10), the printed grid should look like this:

```
.....
j....
...j.
```

Suppose that from this position, the vessel moves north and drops. The new grid should look like this:

```
.....
.....
.....
i....
...i.
```

Notice that the grid had to get taller to ensure that all of the mines are visible and the vessel is still at the center.

If a field has no remaining mines, simply render it as a single dot.

If you need to render a mine that has already been missed, use the `*` (asterisk) character.

Examples

Example 1. Very simple successful run

field file

```
z
```

script file

```
gamma
```

output

```
Step 1

z

gamma

.

pass (5)
```

Example 2. Successful run

field file

```
..Z..
.....
Z...Z
.....
..Z..
```

script file

```
north
delta south
west
gamma east
east
gamma west
south
delta
```

output

```
Step 1


..Z..
.....
Z...Z
.....
..Z..

north

.....
.....
..Y..
.....
Y...Y
.....
..Y..


Step 2

.....
.....
..Y..
.....
Y...Y
.....
..Y..

delta south

.....
.....
X...X
.....
..X..

Step 3

.....
.....
```

```
X...X
.....
..X..
```

west

```
.......
.......
..W...W
.......
....W..
```

Step 4

```
.......
.......
..W...W
.......
....W..
```

gamma east

```
.....
.....
....V
.....
..V..
```

Step 5

```
.....
.....
....V
.....
..V..
```

east

```
...
...
..U
...
U..
```

Step 6

```
...
...
..U
...
U..
```

gamma west

```
.
.
.
.
T
```

Step 7

.
.
.
.
T

south

.
.
S

Step 8

.
.

S

delta

.

pass (8)

Example 3. Complete, but script too long

field file

```
z
```

script file

```
gamma
north alpha
```

output

```
Step 1

z

gamma

.

pass (1)
```

Example 4. Missed a mine

field file

```
..a..
.....
.....
.....
..a..
```

script file

```
north
delta south
south
south
delta
```

output

```
Step 1

a
.
.
.
a

north

.
.
*
.
.
.
*

fail (0)
```

Example 5. Script too short

field file

```
..Z..
.....
Z...Z
.....
..Z..
```

script file

```
north
delta south
west
gamma east
east
gamma west
```

output

```
Step 1

..Z..
.....
Z...Z
.....
..Z..

north

.....
.....
..Y..
.....
Y...Y
.....
..Y..

Step 2

.....
.....
..Y..
.....
Y...Y
.....
..Y..

delta south

.....
.....
X...X
.....
..X..

Step 3

.....
.....
X...X
.....
..X..
```

```
west

.......
.......
..W...W
.......
....W..

Step 4

.......
.......
..W...W
.......
....W..

gamma east

.....
.....
....V
.....
..V..

Step 5

.....
.....
....V
.....
..V..

east

...
...
..U
...
U..

Step 6

...
...
..U
...
U..

gamma west

.
.
.
.
T

fail (0)
```