Problem D
Dungeon Dawdler
Time limit: 3 seconds

Oh no! The wicked wizard Nocoproco has captured you and locked you into his dungeon, from which there is no escape. Looks like you will be here for a while, so you might as well make good use of the time and scout out the place. Fortunately, there are no monsters lurking about in the dungeon, so exploring should be relatively safe. But there are still some complications.

There may be up to two hidden trapdoors in the dungeon. Walking into a trapdoor causes you to fall through it and end up in a different part of the dungeon. To make matters worse, there are few distinguishing features in the dungeon so you have no immediate way of recognizing previously visited locations, and it is easy to get disoriented. However, you do have a keen sense of direction and can always tell which way is north.

The dungeon is a rectangular grid where each cell is either a wall, a plain open space, or a trapdoor. From an open space, the only thing you can discern in the dim light is which of the four adjacent cells (in the directions north, east, south, and west) are walls, and you can then walk to any adjacent cell which is not a wall. If you walk into a trapdoor, you fall into it before you can react, but you have time to observe the four cells adjacent to the trapdoor before falling.

It is possible to go from every location in the dungeon to any other location (but it may require using a trapdoor, as in Sample Interaction 1 below). The dungeon also consists of a single area – if the trapdoors were changed into plain open spaces it would still be possible to go from every location to any other location. The endpoints of the trapdoors may be at any open space in the dungeon, but not on another trapdoor, and the endpoints of the two trapdoors are not at the same position. Your starting point is neither a trapdoor nor an endpoint of a trapdoor.

Write a program to explore the dungeon and create a complete map of it.

Interaction

This is an interactive problem, proceeding in rounds. In each round of interaction, your program first reads information about your current surroundings, and then responds by taking an action.

The information about your current surroundings is given on a single line. The line starts with four characters c_N, c_E, c_S and c_W describing the surroundings of the cell you just walked to (or for the first round, the starting cell), in the directions north, east, south and west respectively. Each character is either '`#`', indicating that the cell in that direction is a wall, or '`.`', indicating that you can walk there. Then on the same line follows either the word "`trap`" if the cell you walked to is a trapdoor, or "`ok`" if it is not. If the cell is a trapdoor, the line contains a third string, describing the surroundings of the endpoint of the trapdoor in the same format as above.

There are two types of actions: moving to an adjacent cell, and reporting that you are done. To move to an adjacent cell, output a single line containing one of '`N`', '`E`', '`S`', and '`W`', corresponding to moving north, east, south, and west respectively. To report that you are done, output a line containing "`done`", followed by a complete map of the dungeon. Output the map as a rectangular grid of minimum size, including all walls seen. First output a line with two integers h and w indicating the height and width of the grid. Then output h lines (from north to south) each containing exactly w characters (from west to east), using the following symbols:

1. '`#`' for walls, and for other unreachable cells that are outside the dungeon.
2. '`S`' for your starting position.
3. '`A`' and '`B`' for the locations of the two trapdoors.
4. '`a`' and '`b`' for the corresponding locations of the endpoints of the two trapdoors.
5. '`.`' for the remaining walkable cells.

It does not matter which of the trapdoors you label ‘A’ and which of them you label ‘B’, as long as ‘a’ is the endpoint of the trapdoor labelled ‘A’, and similarly for ‘b’. If there is only one trapdoor, you may label it either ‘A’ or ‘B’. After outputting the map the interaction ends and your program should exit.

There are at most 500 reachable positions in the dungeon. Your solution may take at most 10^5 steps. If you attempt to walk into a wall, your program will be judged as Wrong Answer. *To facilitate testing of your solutions, we provide a simple tool that you can download from the Kattis page for this problem.* Remember to flush your standard output buffer after each action!

Read **Sample Interaction 1** **Write**

```
#.#. ok
```

```
E
```

```
#.#. trap .###
```

```
N
```

```
##.. trap #.##
```

```
E
```

```
#.#. ok
```

```
E
```

```
#.#. ok
```

```
E
```

```
#.#. trap .###
```

```
done
4 7
#######
#b.SAB#
#####a#
#######
```

Read **Sample Interaction 2** **Write**

```
.##. ok
```

```
W
```

```
..## ok
```

```
N
```

```
#..# ok
```

```
E
```

```
##.. ok
```

```
done
4 4
####
#..#
#.S#
####
```