F Maze Recognition

Problem

You are trapped in a maze that consists of square rooms with doors between some of the adjacent rooms. You do not have a map of the maze. The only things that you know are that there is exactly one exit room (which may be anywhere in the maze), and that the size of the maze will never exceed 100×100 rooms. When you are in a room, you can see in which directions there are doors. If you are in the exit room, you can also clearly see this. You are not only interested in exiting the maze, but you also want to know what the *minimum* number of rooms is that you need to enter before reaching the exit room. Write a program that searches the maze for you, and decides what the shortest possible route would have been.

Input and output

This is a problem with dynamic input and output, read the following description very carefully.

The first line of the input contains a single number: the number of test cases to follow. Then, for every test case the input will depend on the values that your program outputs. The sequence of input and output is as follows:

1. Input: One line with a string containing the possible moves that you can make in the current room. These moves can be any subset of `'N'`, `'E'`, `'W'` or `'S'` (for North, East, West or South respectively), in this order. There will always be at least one possible move. If the current room is the exit room, the string ends with a single `'*'`.
2. Output: Your output, on a single line, must then be one of the following:
   (a) A character from `'N'`, `'E'`, `'W'`, `'S'`, indicating that you make one of the possible moves. After this, go back to 1 (in the new room).
   (b) An integer number which is your answer for this test case. This must be the minimum possible number of rooms you need to enter before reaching the exit room, or -1 if the exit room cannot be reached. This ends the sequence of input and output for this test case.

If your program outputs an invalid move or any other invalid output, you will receive a Wrong Answer.

Notes

C/C++ users:

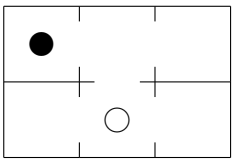
- Do `fflush(NULL);` after you print output.
- If you use `scanf` to read the input, then do not read newline characters. So, to read an integer n with `scanf`, use `scanf("%d", &n);`.

Java users:

- Do not use *buffered* output, `System.out.println` will do fine.

Example

Let us for example consider the following maze. The open circle represents the starting room, the filled circle represents the exit room.



Note that there might be multiple ways to find the correct answer, the sequence below is just an example. Extra newlines in the example below are for clarity only; you should print exactly one newline character after each line of output.

Input	Output
`1` `NEW`	
	`N`
`EWS`	
	`E`
`W`	
	`W`
`EWS`	
	`W`
`E*`	
	`2`