Problem 6—Puzzle

A children's puzzle that was popular when I was a kid consisted of a 5x5 frame which contained 24 small squares of equal size. A unique letter of the alphabet was printed on each small square. Since there were only 24 squares within the frame, the frame also contained an empty position which was the same size as a small square. A square could be moved into that empty position if it were immediately to the right, to the left, above, or below the empty position. The object of the puzzle was to slide squares into the empty position so that the frame displayed the letters in alphabetical order.

Here is a puzzle in one particular configuration:

```
T R G S J
X D O K I
M   V L N
W P A B E
U Q H C F
```

Now, if we let L, R, A, and B represent sliding the square that is situated to the left, to the right, above, and below the empty position respectively, we could designate a series of moves such as: ARRBBL. After making these moves, the puzzle looks like this:

```
T R G S J
X O K L I
M D V B N
W P   A E
U Q H C F
```

You are to write a program simulating the operation of this puzzle.

INPUT SPECIFICATION

The input consists of several puzzles. Each is described by its initial configuration and the sequence of moves on the puzzle. The first 5 lines of each puzzle description are the starting configuration. Subsequent lines give the sequence of moves.

The first line of the input case corresponds to the top line of squares in the puzzle. The other lines follow in order. The empty position in a frame is indicated by a space. The first lines have 5 characters each (and each terminated by **<EOLN>**). Of these 25 characters, exactly one is a space. All others are capital letters.

The next line or lines contain the sequence of moves to be made on the puzzle. A sequence of moves is an arbitrary sequence of the letters A, B, L, and R, followed by zero and **<EOLN>**. The zero indicates the end of the sequence of moves; it will be followed immediately by **<EOLN>** and the next case will follow immediately after that. The sequence of moves itself need not all lie on the same line. Any number of **<EOLN>**'s may appear before, after, or between moves in the sequence. However, moves will not be separated by any other character, including spaces. The only spaces that will appear in the input correspond to empty spaces in the initial configurations of the puzzle.

Zero followed by **<EOLN>** appears immediately after the last test case in the input file. This will signify the end of the input data.

OUTPUT SPECIFICATION

The test cases should appear in the output in the order in which they appear in the input. The output for each test case begins with `Puzzle #n:`**<EOLN>**, where n represents the number of the puzzle in the input file, starting from 1. If the sequence of moves specified in this test case should result in an error, the following line should appear in the output:
`This puzzle has no final configuration.`**<EOLN>**

If the puzzle should have a valid final configuration, this configuration should be printed in 5 rows of 5 characters. The empty square should be printed as a space. An extra space should be printed immediately after each of these 25 characters, and **<EOLN>** should terminate each row.

An extra **<EOLN>** should follow each test case in the output.

SAMPLE INPUT

```
TRGSJ<EOLN>
XDOKI<EOLN>
M VLN<EOLN>
WPABE<EOLN>
UQHCF<EOLN>
ARRBBL0<EOLN>
ABCDE<EOLN>
FGHIJ<EOLN>
KLMNO<EOLN>
PQRS <EOLN>
TUVWX<EOLN>
AAA<EOLN>
LLLL0<EOLN>
ABCDE<EOLN>
FGHIJ<EOLN>
KLMNO<EOLN>
PQRS <EOLN>
TUVWX<EOLN>
AAAAABBRRRLL0<EOLN>
0<EOLN>
<EOF>
```

SAMPLE OUTPUT

```
Puzzle #1:<EOLN>
T R G S J <EOLN>
X O K L I <EOLN>
M D V B N <EOLN>
W P   A E <EOLN>
U Q H C F <EOLN>
<EOLN>
Puzzle #2:<EOLN>
  A B C D <EOLN>
F G H I E <EOLN>
K L M N J <EOLN>
P Q R S O <EOLN>
T U V W X <EOLN>
<EOLN>
Puzzle #3:<EOLN>
This puzzle has no final configuration.<EOLN>
<EOLN>
<EOF>
```