There was once a 3 by 3 by 3 cube built of 27 smaller cubes. It has fallen apart into seven pieces:

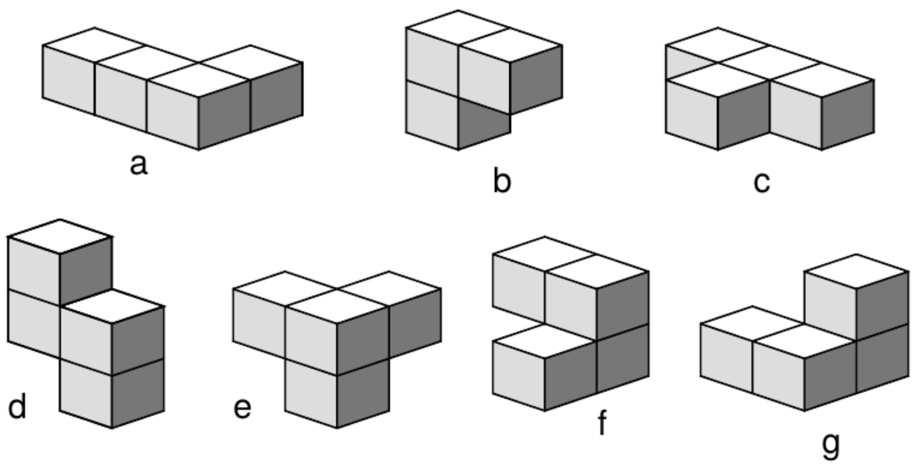

Figure 1: The seven pieces that once formed a cube

The seven pieces can be assembled in many ways to again form the cube. Figure 2 shows two of these possibilities. The first square stands for the front plane, the next one for the middle plane and the last one for the back plane of the cube. The letters in the cells stand for the name of piece filling out the corresponding space in the cube. The name of the seven pieces can be found in Figure 1.

a	d	c
a	c	c
a	a	c

d	d	g
b	f	g
f	f	e

d	g	g
b	f	e
b	e	e

a	a	b
a	b	b
a	d	c

f	f	e
g	f	c
d	d	c

f	e	e
g	g	e
d	g	c

Figure 2: Two possibilities of assembling the cube

You are to write a program that outputs all possibilities of assembling the cube but suppress solutions that are mere rotations of another solution.

Input

The input file has several test cases. Each test case indicates the initial position of piece '`a`'. You can translate it, but you mustn't rotate it.

Output

For each solution found, your program should output a line containing the solution as a string. The string is a linearized form of the cube. Each letter stands for the piece filling out the corresponding space in the cube. It is linearized as follows:

- The string consists of substrings representing the front, middle and back plane.
- Each substring consists of substrings representing the top, middle and bottom row.
- Each row substring consists of letters representing the left, middle and right cell.

The solutions in Figure 2 would be represented like this:

```
adcaccaacddgbfgffedggbfebee
aababbadcffegfcddcfeeggedgc
```

It is very important that your program uses the naming convention given in Figure 1 and linearizes the cube as explained above.

Print a blank line after each test case.

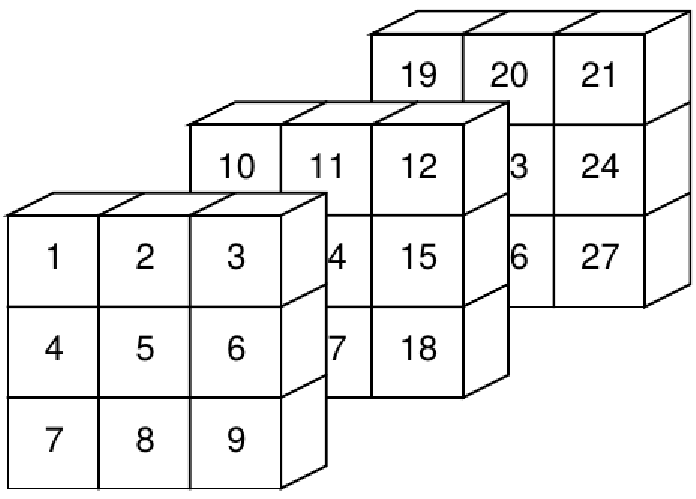

Figure 3: Positions of the cells in the string

Figure 3 again shows how the cells of the cube are linearized.

Hint: Piece '`a`' is the only part that, by rotation and translation, cannot be transformed into itself. In order to avoid solutions that are mere rotations of an already found solution, you may restrict transformations of piece '`a`' to translations.

Sample Input

```
aa.a..a....................
.........a..a..aa.........
```

Sample Output

```
aababbadcggeffcddcgeegfedfc
aababbadceffgdcgdceefedfggc
...
aababbadcffegfcddcfeeggedgc

adcaccaacfddfebgeeffdggbgeb
...
```