The rotation game uses a # shaped board, which can hold 24 pieces of square blocks (see Fig.1). The blocks are marked with symbols 1, 2 and 3, with exactly 8 pieces of each kind.

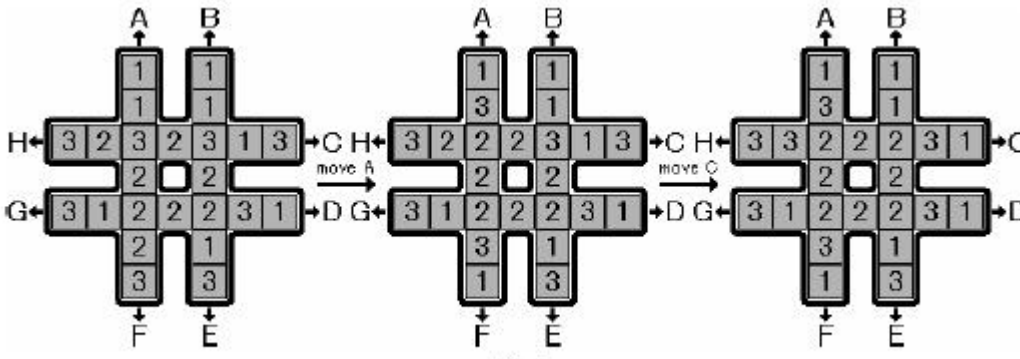

Fig.1

Initially, the blocks are placed on the board randomly. Your task is to move the blocks so that the eight blocks placed in the center square have the same symbol marked. There is only one type of valid move, which is to rotate one of the four lines, each consisting of seven blocks. That is, six blocks in the line are moved towards the head by one block and the head block is moved to the end of the line. The eight possible moves are marked with capital letters **A** to **H**. Figure 1 illustrates two consecutive moves, move **A** and move **C** from some initial configuration.

Input

The input consists of no more than 30 test cases. Each test case has only one line that contains 24 numbers, which are the symbols of the blocks in the initial configuration. The rows of blocks are listed from top to bottom. For each row the blocks are listed from left to right. The numbers are separated by spaces. For example, the first test case in the sample input corresponds to the initial configuration in Fig.1. There are no blank lines between cases. There is a line containing a single '`0`' after the last test case that ends the input.

Output

For each test case, you must output two lines. The first line contains all the moves needed to reach the final configuration. Each move is a letter, ranging from '`A`' to '`H`', and there should not be any spaces between the letters in the line. If no moves are needed, output '`No moves needed`' instead. In the second line, you must output the symbol of the blocks in the center square after these moves. If there are several possible solutions, you must output the one that uses the least number of moves. If there is still more than one possible solution, you must output the solution that is smallest in dictionary order for the letters of the moves. There is no need to output blank lines between cases.

Sample Input

```
1 1 1 1 3 2 3 2 3 1 3 2 2 3 1 2 2 2 3 1 2 1 3 3
1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3
0
```

Sample Output

```
AC
2
DDHH
2
```