PSH: ^ ! (aka Hidden Constants and Underlying Data Structures Matter)

In the game `^ !` (pronounced with enthusiasm "Caret!"), a player tries to transform a initial 4 x 5 pattern of boolean values to a target pattern.

In order to make a transformation, the player chooses a pair of adjacent values x and y and changes their values to x' and y' as follows:

$x' = x \wedge y$

$y' = !\, x$

where

- x is either the left value in a pair of horizontal values, or the top value in a pair of vertical values
- y is either the right value in a pair of horizontal values, or the bottom value in a pair of vertical values
- x' and y' are the new values for the cells that contained x and y, respectively
- ^ is the *exclusive or* operator
- ! is the *not* operator

Input Format

Input starts with the initial pattern specified in 4 rows of five characters. Each character will either be an `f` indicating a false value, or a `t` indicating a true value.

There there is a blank line, followed by the target pattern specified in the same way as the initial pattern.

Constraints

The input will conform to the input format specification. You should not do any input validation.

Your first solution must use the Board class that is given. You may add additional methods to this class or override inherited ones. You must use at least 3 of the following classes/interfaces from the Java Collections Framework: ArrayList, ArrayDequeue, HashMap, HashSet, LinkedList, Map, PriorityQueue, Queue, Set, Stack, TreeSet, and TreeMap. In addition to meeting these constraints, the goal for this solution is to create one that is correct and fast enough to pass 10 test cases. If you meet these constraints, this solution would be worth 80 points.

Your second solution must improve on the time of your first solution. For this solution, you may change the Board class and use your own implementations for underlying data structures. If you improve on your previous time, and you pass test case 10, you will earn 10 additional points. You can earn additional points by passing the remaining test cases. Up to 5 points of extra credit will be awarded to the five fastest solutions.

Output Format

The first line of output should contain the minimum number of moves needed to transform the initial pattern to the target pattern.

The remainder of output will show the pattern after each move. There should be a blank line before each of these patterns.

Note: Most of the time there are multiple sequences that will transform the initial pattern to the final pattern using a minimal number of moves. To break a tie in the sequences, use the following procedure.

For patterns with the same initial sequence of moves, use the following tie breaking rules, applied in order, to choose the next move:

1) Choose horizontal transformations before vertical transformations.

2) If still tied, use the transformation applied to the higher row over one applied to a lower row.

3) If still tied, use the transformation applied to the leftmost column.

Sample Input 0

```
fffft
tttft
ttttf
ttftt

fftft
tttft
ttttf
ttftt
```

Sample Output 0

```
1

fftft
tttft
ttttf
ttftt
```

Sample Input 1

```
ttttt
ttttt
ttttt
ttttt

fftft
tffft
ftttf
ftttf
```

Sample Output 1

```
5

ffttt
ttttt
ttttt
ttttt

ffttt
tfftt
ttttt
ttttt

fftft
tffft
ttttt
ttttt

fftft
tffft
ftttt
ftttt

fftft
tffft
ftttf
ftttf
```

```
```

Sample Input 2

```
fttft
ttttf
tfttf
tfftf

fftff
tfttt
tffff
tfttf
```

Sample Output 2

```
6

ftttt
ttttf
tfttf
tfftf

fttff
ttttf
tfttf
tfftf

fttff
ttttf
tffff
tfftf

fttff
ttttf
tffff
tfttf

fftff
tfttf
tffff
tfttf

fftff
tfttt
tffff
tfttf
```