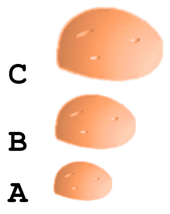
Problem 14: Flipping Pancakes[1]

Source filename:	`pancakes.(cpp|java)`
Input filename:	`pancakes.in`
Output filename:	`pancakes.out`

Alice owns a nice restaurant. Although her restaurant serves 3 meals a day, Alice is especially proud of the quality of the breakfast that her restaurant serves. Unfortunately, her normal chef is sick and will be unable to work for a couple of weeks. Alice has hired a temporary chef who for the most part is doing OK. However, he is not very good at cooking pancakes. Since Alice's Restaurant is famous for its pancakes, this shortcoming is causing Alice a great deal of consternation.

Here are the temporary chef's problems. The sizes (i.e., diameters) of the pancakes have any one of six different diameters; and he ALWAYS burns exactly one side of every pancake. Having been a marketing major in college, Alice is trying to make the best of a less than ideal situation until her regular chef can return to work. She reasons that the customers may not notice the problems with the pancakes if she arranges the stacks in the following manner. Using a spatula, Alice flips a stack of pancakes several times until all of the burned sides are facing downward. In addition, she flips the stack in such a way that no pancake is on top of a pancake of larger diameter. This way the larger pancakes tend to hide the smaller ones. Obviously, Alice is very handy with a spatula. Using one hand she can insert the spatula underneath any pancake in a stack of pancakes and flip every pancake above the spatula without dropping a single one on the floor.

For the sake of description, we'll label the six pancake diameters with the letters **`A`** through **`F`**, where **`A`** represents the smallest pancake and **`F`** the largest. Thus, a stack of 3 pancakes could be represented by the letters:



Rather than list the pancake sizes in the vertical format shown above, for convenience sake, we'll list them horizontally – with the understanding that the first letter listed corresponds to the size of the pancake located at the *bottom* of the stack. Thus, the stack shown above will now be represented by the string **`ABC`**.

[1] The problem appeared on a contest sponsored by the MidSouth College Computing Conference, April 2, 2005.

In addition, we'll use lower-case letters to represent pancakes that are stacked with the burned side up. For example, the stack **`cbA`** contains 3 pancakes arranged with the largest pancake on the bottom, the 2nd largest in the middle, and the smallest is on the top. In addition, all but the top pancake have their burned side facing up. Notice that in this case, Alice could "fix" the stack by first flipping the top pancake, and then flipping all three. The resulting stack would be in the configuration **`ABC`**, which meets Alice's marketing strategy.

Any configuration of pancakes can be altered to meet the marketing guidelines by performing a series of flips. For example, the stack **`abc`** can reach the goal by the following series: flip all 3 (yields **`CBA`**), flip top 1 (**`CBa`**), flip all 3 (**`Abc`**), flip top 2 (**`ACB`**), flip top 1 (**`ACb`**), flip top 2 (**`ABc`**), flip top 1 (**`ABC`**). However, this series of 7 flips isn't optimal. The following series of 6 flips will also yield the desired result: flip 2 (**`aCB`**), flip 3 (**`bcA`**), flip 2 (**`baC`**), flip 3 (**`cAB`**), flip 2 (**`cba`**), flip 3 (**`ABC`**).

Write a program that will find and report the *minimum* number of flips needed to rearrange any stack of pancakes to meet Alice's marketing strategy. A stack may contain from 1 to 6 pancakes and will be represented by a character string of 1 to 6 letters. These will be upper- or lower-case letters between **`A`** and **`F`**, inclusive.

The input file contains 1 or more lines. Each input line, except the last line, contains a string of 1 to 6 upper- or lower-case letters between **`A`** and **`F`**, inclusive. The last line contains the digit **`0`** and is used to denote the end of input file. There are no spaces in the input file.

For each input string, your program should determine the minimum number of flips needed to arrange the stack of pancakes in the proper order (no pancake is on top of a pancake of larger diameter) and with all burned sides facing down. Your program should print this minimum number to the output file, one number per line.

Read input from the file named **`pancakes.in`** and write output to the file named **`pancakes.out`**. All output to the screen will be ignored.

Example Input	Example Output
`A`	**`0`**
`a`	**`1`**
`ba`	**`1`**
`ab`	**`4`**
`abc`	**`6`**
`aaa`	**`1`**
`fca`	**`1`**
`ABC`	**`0`**
`0`	