Disorder

Computer Science Society
Programming Contest
Spring 2010

There's more than one way to sort a list of character strings. In *dictionary* order, strings are sorted primarily by first character, secondarily by second character, and so on; if two strings have different lengths and match on all of the characters in the shorter string, then the shorter string precedes the longer string. In *lexicographical* order, strings are sorted primarily by length (shorter strings precede longer strings), secondarily by dictionary order. In this problem, we are given an unsorted list of strings without duplicates, and are asked to compute the distance between the list sorted in dictionary order and the list sorted in lexicographical order. Since the same strings appear in both lists and there are no duplicates, we can look at a string and measure the difference between the indices at which it appears in the two sorted lists. The *distance* between the two sorted lists is the sum of those differences over all words in the original list.

Input Format

The input consists of one or more *lists*, each consisting of one or more nonempty input lines followed by an empty input line. The characters on a nonempty input line constitute a *string*. All of the strings within a list are distinct and contain only ASCII printable characters (no blanks).

Output Format

For each list of strings in the input, compute and output the distance between the list sorted in dictionary order and the list sorted in lexicographical order. Warning: what the Java API specification calls "lexicographic ordering" (e.g. in the `String.compareTo` method) is what I call dictionary order.

Input Sample

```
what
is
going
on
with
this
list

foo
foobar
food
fooey

weather
cat
problem
father
at
date
```

Output Sample

```
12
4
0
```