Problem 2: Word Frequencies

In this problem, you will implement a program to find all words in the input and print them in a sorted order, followed by their frequencies in the text. We will consider words to be any sequences of letters, uppercase or lowercase. You should use the Scanner class to read the input, declaring a scanner using a line like this, as usual:

```
Scanner sc = new Scanner( System.in );
```

To detect the end of input, you can use the Scanner method `hasNext`, and you should use the method `next` to read the words. However, the default behaviour of these methods is such that they would include any visible characters, including digits and punctuation, into words (i.e., word tokens). Scanner allows us to modify its behaviour by specifing a different delimiter pattern; i.e., a pattern that describes what kind of strings separates words. To achieve the desired behaviour, you will need to use the following line:

```
sc.useDelimiter( "[^a-zA-Z]+" );
```

which will make our scanner `sc` ignore everything other than letters, and use only words consisting of letters.

When you implement a program that reads input using the methods `next` and detecting the end using the method `hasNext`, a problem will be how to indicate the end of input to the program when you use the keyboard. On a Linux or Mac system you will need to type Ctrl+d key combination (press key Ctrl and while holding press key 'd'), while on Windows you need to press Ctrl+z.

All words should be translated into lowercase letters and you can use the method `toLowerCase` for strings to achieve this. For example, given a string variable `s`, the following expression returns the string `s` in lower-case letters:

```
s.toLowerCase()
```

As a step in processing, you will need to sort all words. It is guaranteed that the input will not have more than 10000 words, so you if you allocate an array of 10000 strings, you are guaranteed to store all words there. You should implement one of the sorting algorithms used in class to sort the words. Strings are compared in java using the `compareTo` method. You can read about it in the textbook.

Input

The input consists of arbitrary text. If you are entering input from the keyboard, you will need to press Ctrl+d or Ctrl+z to signal the end of input. Otherwise, you can feed input from a file in a command-line window using:

```
java Problem2 < file.txt
```

for example. It is assumed that the input does not have more than 10000 words.

Output

In the output, your program must print all words, sorted alphabetically, one word per line followed by its frequency, as shown in the example.

Sample Input	*Sample Output*
3 This is a test. abc test this is; is a test test test c d d c d d d d the last line	a 2 abc 1 c 2 d 6 is 3 last 1 line 1 test 5 the 1 this 2