**Hashing lab assignment**

The Shakespeare play *Romeo and Juliet* contains 23,869 words of dialog, but only 3,684 unique words. This dialog, stripped of all punctuation, is available on mulan as the file [RomeoAndJuliet.txt](http://mulan.csufresno.edu/%7Etwilson/RomeoAndJuliet.txt). ut some statistics about the efficiency of the hashing process.

 Here are the specifics.

Your program will consist of a single file, rjhash.cpp.

You should declare a constant HASHSIZE = 4001 -- i.e., a prime number more than enough to hold all the unique words in the play -- and then implement a simplified hash-array class along the lines of our textbook, where the array itself has size HASHSIZE and value type int. This int value will be used to count occurrences of words in the text, and will be initialized to 0 when the array is created. The collision-resolution technique is linear probing, with

*p*(*k*,*i*)=*i*

.

The hash array class should

* keep track of the number of elements in the hash array, and provide a method size() that returns this value
* provide an update(key) method that inserts key into the hash array if it doesn't already appear (setting its value to 1), or increments the value associated with key in the hash array if it is already there
* keep track of the number of array accesses that were required for the previous call to update, and provide a method probes() that returns this value (this value will be overwritten with each new call to update)
* provide reset() and next(key&,val&) methods to navigate the contents of the hash array; reset sets the "current" element to be -1, and next sets it to the next non-empty element of the array greater than current and returns this element's key and value (or the empty string and -1, if there are no more elements) in reference parameters.

The program should read words from its standard input until EOF and update the hash array accordingly, keeping statistics as it goes along. The statistics are stored in two arrays of size HASHSIZE:

* updates[i] keeps track of the number of updates that were done while the size of the hash array is equal to i
* cost[i] keeps track of the total number of probes that were done while the size of the hash array is equal to i

The quotient of these two numbers represents a good estimate of the expected number of probes needed when the hash array has i elements in it.

Finally, your program should output to standard output:

* the number of unique words in the input file (i.e., the size of the hash array when the input has all been read),
* a comma-separated list of the ratios cost[i]/updates[i], each to two decimal places, where i goes from 1 to the size of the hash array

This comma-separated list could then be easily read into a spreadsheet and a graph produced that would look something like Figure 9.10 in the PDF textbook (with just one curve, though).

For some extra credit, also output a list of the 24 most often occurring words in the play and the numbers of times they occur.