Computer Systems (SS 2009)
Exercise 2: April 7, 2009

Wolfgang Schreiner
Research Institute for Symbolic Computation (RISC)
Wolfgang.Schreiner@risc.uni-linz.ac.at

March 12, 2009

The exercise is to be submitted by the denoted deadline via the submission interface of the Moodle course as a single file in zip (`.zip`) or tarred gzip (`.tgz`) format which contains the following files:

- A PDF file `Exercise`*`Number`*`-`*`MatNr`*`.pdf` (where *Number* is the number of the exercise and *MatNr* is your "Matrikelnummer") which consists of the following parts:
  1. A decent cover page with the title of the course, the number of the exercise, and the author of the solution (identified by name, Matrikelnummer and email address).
  2. For every source file, a listing in a *fixed width font*, e.g. `Courier`, (such that indentations are appropriately preserved) and an appropriate *font size* such that source code lines to not break.
  3. A description of all tests performed (copies of program inputs and program outputs) explicitly highlighting, if some test produces an unexpected result.
  4. Any additional explanation you would like to give. In particular, if your solution has unwanted problems or bugs, please document these explicitly (you will get more credit for such solutions).
- Each source file of your solution (no object files or executables).

Please obey the coding style recommendations posted on the course site.

Exercise 2: Text Statistics

Write a program `TextStatistics` N that takes a natural number `N` as its command line argument, reads a text from the standard input and prints to the standard output

- the total number of words in the text,
- the number of different words in the text,
- the arithmetic mean of the number of occurrences of a word,
- the median of the number of occurrences[1],
- the standard deviation of the number of occurrences,
- the N words that occur most frequently in the text (together with the number of their occurrences).

The output format shall be for input "word, WORD, new, ... another!" (close to) the following:

```
Total number of words:                          4
Number of different words:                      3
Arithmetic mean of number of occurrences:       1.33...
Median of the number of occurrences:            1
Standard deviation of the number of occurrences: 0.40...
The words that occur most frequently are:
       2:     word
       1:     new
```

As a "word" we consider any sequence of characters that only consists of (upper and lower-case) letters in the ASCII alphabet; for the purpose of above statistics we do not distinguish between upper and lower case letters, i.e. "word" and "WORD" denote the same word.

The program is to be based on a class `Text` with the following operations:

```
Text text(5);            // text uses hash table of length 5
text.enter("word");      // enter word into text
```

[1] `http://en.wikipedia.org/wiki/Median`

```
text.enter("WORD");      // enter same word again
text.enter("new");       // enter a new word
text.enter("another");   // another new word
text.done();             // no more word will be entered

cout << text.getTotalNumber(); // 4 words in total
cout << text.getDiffNumber();  // 3 different words
cout << text.getMean();        // 1.33 = 4/3
cout << text.getMedian();      // occurrences 1 1 2 -> median 1
cout << text.getStdDev();      // sqrt((2*(1-1.33)^2 + 1*(2-1.33)^2)/4)

// argument must be less than or equal the number of different words
string *a = text.getMostWords(2); // a[0] = "word", a[1] = "new"
int *b = text.getMostNumber(2);   // b[0] = 2, b[1] = 1
```

The main program allocates a text object, reads every line from the standard input stream, decomposes the lines into words, enters the words into the text object, indicates to the object the end of the text, and then determines by the object's member functions the appropriate results.

A text object created by a constructor call `Text(`n`)` is represented by a *hash table* (an array) of size n where each element of the table is a linked list of nodes; each node holds a word and the number of its occurrences (a constructur call `Text()` shall be also possible with a default value for n).

Write auxiliary classes `WordList` *and* `WordNode` *for a list and its nodes with appropriate member functions for performing the required operations.*

When a word w is entered into the text object, it is *hashed* to a unique position i in the table and the list at that position is investigated. If w occurs in the list, the number of occurrences is increased by one, otherwise a new list node is created that holds w with occurrence 1.

Do not invent your own hash function, C++ implementations of various hash functions that map strings to numbers can be found at the "General Purpose Hash Function Algorithms" site[2].

When the member function `done` is called, all list nodes are transferred to an array which is sorted according to the number of occurrences.

Sort the array in place by the Quicksort *algorithm for which plenty of resources are publically available (within the JKU network, you have access to*

[2] `http://www.partow.net/programming/hashfunctions`

the original publications[3]*).*

Using this array, the member functions `getMean` and `getMost` can be easily implemented.

Test your program with various test cases that include an empty text, a one word text, and the sample provided in the Moodle assignment for $N = 25$.

Note: in most operating systems, you can execute in a command shell

```
./TextStatistics 25 < sample.txt
```

to redirect the content of file `sample.txt` to the standard input of the program call `TextStatistics 25` (with the executable file `TextStatistics` residing in the current working directory).

[3] *Alg. 64: Quicksort* `http://portal.acm.org/citation.cfm?id=366622.366644`, *Alg. 63: Partition*, `http://portal.acm.org/citation.cfm?id=366622.366642`, C.A.R. Hoare, Communications of the ACM, 1961, Volume 4, Issue 7, July 1961.