Assignment 01: iCountApp

Objectives	Submission and due date
This is your first real programming assignment in data structures and algorithms. Start early and you will enjoy it. The objectives are: 1. Begin to work with C/C++; 2. Build tools for large dataset computations; 3. Learn about pointers; 4. Design and implement your first data structure; 5. Implement your first search algorithm; 6. Experience with linked lists.	COMP305: WebCT — Assignment 01 Due: 23:00 — 15 Mar. 2013 Instructor: — George Baciu Title: — iCountApp Total Marks: — 20 Note: — Individual work

Description

In this assignment you will write a simple program that will read an arbitrary text file, count the frequency of occurrence of each word that is found in the file, **and sort the output** in the decreasing order of count. For example, let the file in your "dat" directory be "dat/data.txt." Suppose this file contains the following text:

Marry had a little lamb a little kitten and a little bunny

Assume that the text doe not contain any punctuation symbols, i.e . ; : =- etc. By running your program in cygwin from the src folder as follows:

$ ../bin/main ../dat/data.txt

The output generated should be:

No	Word	Count
1	a	3
2	little	3
3	Marry	1
4	had	1
5	lamb	1
6	kitten	1
7	and	1
8	bunny	1

Please note that each word in the output is unique. That means that, as your program reads in the data file, it stores the words in the data structure that you will implement. Then, the program will search for each new word to find if the word has been read before. If not, it will create a new entry with a count of 1. If yes, then it will increase the word count by 1.

The important part of your program is the data structure that will support an arbitrarily long text file. This file could potentially be 10GB or larger. So, you cannot use a fixed size array as you do not know in advance how big the file is. Hence, you must implement a linked list as your first dynamic data structure. As you learn more about other data structures, in the future you will be able to replace this data structure with a more efficient non-linear data structure that will support searching much faster. But, for now use a linked list implementation.

Your linked list will contain nodes with two fields, a ***word*** field and a ***count*** field. You will first build the linked list; then, you will code functions (or operators) that will operate on the linked list:

No	List Operation	Description
1	listCreate(theList)	This function allocates space for the header of the list and initializes the "theList" as the header. You may have a list record "theList" that manages the entire list. This could contain pointers such as the "head" and the "tail" pointers as well as the number of nodes or node count.
2	listAdd(word, theList)	This function is special; it should add the word only if it is not already in the list and it should do the count computation in order to count the frequency of the words found.
3	listSearch(word, theList)	This function should return true if the word is found in theList, false otherwise.
4	listPrint(theList)	This function should print the entire list pointed to by "theList"
5	listDelete(theList)	This function should de-allocate the space taken by all nodes so that the program can gracefully return the space back to the operating system.

User Interface

The input should be read from a text file from anywhere in the system. That means, you should not code the file name inside the program. Rather, you should use the parameters to the main function in order to get the file name. Examples of running the program in the cygwin window (all should run) are as follows:

```
$ main          ../dat/data.txt    # run from the bin directory (or folder)
$ bin/main      dat/data.txt       # run from parent of bin directory
$ ../bin/main   ../dat/data.txt    # run from the source src directory
```

Note: you will be able to retrieve the file name from the command line in the parameter argv in function main. *argv[1]* will contain the string of the entire file name given on the input line. You can use this parameter to try to open the file.

Packaging and Submission

Your assignment folder should be named by your student user id. Example, in cygwin, on the J: drive (in the COMP network), you should have the following tree structure for your program:

```
$ comp305b
        asg
            asg01
                10887788d
                    bin
                        main.exe    # executable program
                    dat
                        data.txt    # data file in text format
                    src
                        main.cpp    # source code for C/C++
                        makefile    # makefile for compiling the program
            asg02
```

The important part is the student id folder "**10887788d.**" This contains the entire system development tree and the source code of your program. When you hand in the assignment you should zip this folder and submit it to WebCT as "**10887788d.zip**" Please conform to this directory structure and submission archive, otherwise we will not compile and run your program and you will not get marks for the assignment!

Evaluation

[5/20]	Documentation and programming style, ie internal and external user documentation; please see the examples in the doc folder of the lab sessions.
[15/20]	Technical correctness and originality; Please note a real linked list will be required in the implementation. If you use arrays it will not work when the input data is very large. Also you should check for buffer overflow in inputting character strings.

References

Lectures 1-4 http://www.comp.polyu.edu.hk/~csgeorge/comp305/lec/

Labs 1-4 http://www.comp.polyu.edu.hk/~csgeorge/comp305/lab/

Good luck!