Homework #4 - Linked Lists Module

Objective: Implement a linked list abstract data type as a C module. Practice the use of pointers and memory allocation. Compare the time efficiency of linked lists against dynamic arrays.

Background: For a review on linked lists study G. Semeraro "Chapter 4: Data Structure" and also N. Parlante "Linked List Basics" (posted in MyCourses)

Program Development and Testing:

1. Upload the file **hw4_files.tar** (available in MyCourses) to your **hw4** working directory. The tarball contains the:

   header file **LinkedLists.h & ClassErrors.h**

   C framework files: **LinkedLists.c TestList.c**

   Some simple test files: **oneTest.txt, twoTest.txt and fiveTest.txt**

2. Implement a ***linked list module*** using the interface specification in the file **LinkedLists.h** and **.C**. In this assignment the list will store words from a dictionary so you should define the structure **ElementStructs** such that it has two fields: an integer filed, to hold the position of the word in the input file, and a character–string field to store the word itself (like in Homework #2).

3. Add and use **MALLOC_DEBUG(P)** and **FREE_DEBUG(P)** debug macros to the **ClassErrors.h** file. After each malloc() and free() call in your linked list code, call the corresponding macro to print tracking message to stderr. Each macro should produce output of the form:

   ```
   "debug: malloc() of <pointer value> at line <number> in <file name>"
   "debug: free()   of <pointer value> at line <number> in <file name>"
   ```

   The terms between the"<>" should be replace with the appropriate values. The macros will be enabled using "-DMDEBUG" on the compiler line. You can debug your malloc()/free() calls using these macros. You will use the **ClassErrors.h** file in all your future assignments. Your makefile must have this macros disabled.

4. Write a test program, call it **TestList.c**, to test the module implementation of all the functions given in the header file **LinkedLists.h**. For development testing, use **oneTest.txt**, **twoTest.txt** and **fiveTest.txt**. For your final testing and makefile, use the file **us-eng-words.txt** from HW 2 as input. Your output must be origin 1. The calling syntax should be:

   ./TestList us-eng-words.txt

5. Once you tested the basic functionality of your double linked list you will add one more function, called **SearchList(...)**, *to search the linked list* for a word. Write a separate program to test the search function and call it **TestSearch.c**. Your output must be origin 1. The calling syntax should be:

   ./TestSearch us-eng-words.txt word

   Where: **word -** is the word to search for.

Program Behavior:

a. Your C programs should get their *input from a file* whose name is passed on the *command-line*.
b. The TestList programs (only) should *print the first and last 6 elements* of the data set, and the *total number of words* in the input file, to standard output.
c. If any program is called with a wrong number of arguments, the programs should print a friendly usage message to **stderr** and return an error code equal to 1. The friendly message should include a brief description of the program, the syntax and the meaning of key options.
d. If the input file could not be opened, your programs should print a friendly error message to **stderr** and return an error code equal to 2. The message should include the name of the file you were trying open.
e. If the programs complete successfully they should return an error code of 0.
f. The front of a list is defined as the first word in the text file. All output message must be origin one.

Timing against Dynamic Arrays:

1. Write another program to compare the time efficiency of the linked list versus the dynamic array (that you implemented in hw#2), call this program **TestTime.c**. The compilation line will look something like this:

   gcc -Wall -std=c99 -pedantic -DEN_TIME TestTime.c DynamicArrays.c LinkedLists.c -o TestTime

   You should time, independently, data I/O (reading the dictionary) and searching (use the word "space"). Record the timings by redirecting the input to a file called **time.txt.**

   You should reuse your Timing.h marcros. Depending on how you implemented your macros, you may find that you get: "warning: ISO C90 forbids mixed declarations and code", you must fix this problem.

   Note: To compare searching for a word you must also add the function: **SearchArray(...)** to your DynamicArrays module.

Analysis:

Write an **analysis.txt** summarizing your implementation and timing results. Create a tarball **lastName_hw4.tar** (lastName is your last name) with all relevant files and submit it.

Makefile:

You must provide a quality Makefile with the following targets: all, test, search, mem, time, help and clean.

"all"	-should make TestSearch, TestList and TestTime
"test"	- should run **TestList** with output redirected to **out.txt**
"search"	- should run **TestSearch** with make macro variable of **SEARCHWORD**, the word should be "**space**". Output should be redirected to "**search.txt**"
"time"	- should run **TestTime** with the same parameters as search but directed into **time.txt**
"mem"	**-** should run the test options with **valgrind** redirected to **mem.txt**
help, clean	- should do the normal things

Grading Criteria

1. (40 points) Correct implementation of basic Linked List Module, including memory leaks.
2. (20 points) Correct implementation of search function.
3. (10 points) Correct timing test.
4. (10 points) Correct program behavior (error messages, etc)
5. (10 points) Correct make file
6. (10 points) Analysis of results concise and clear.

Notes:

1. To learn more about doubly linked lists read chapter 4 of G. Semeraro's book (posted onMyCourses). If you use other reference sources list those in your **analysis.txt** file.