Comp305 – data structure

Assignment 01 – "iCountApp"

The guide aims at highlighting the important points and requirements and providing some guides for you to jump start doing the assignment and aspiring a **higher** marks, by making less careless mistakes, avoiding overlooking the requirements and attending to better programming style and documentation standard.

Points to note:

- Follow closely the directory tree structure of the "submission and package" requirements in the assignment 01 file
- Take the filename from the main parameter, i.e. should not run the program as follows: main < ../dat/data.txt ; Use *argv[1]* in the *main()* parameter as suggested in the assign01 file.
- If you use tools other than *cgwin* to develop your program, you need to recompile your program in the *cgwin* environment, to make sure it works under *cgwin* . Since we may not recomplile your program, and you will be unnecessarily lost marks because if your program does not run under *cywin* environment.
- Programming style: *(refer to the programs in the lab)*
    - Structural: Design proper function() call from main(); Put constants in the beginning of the program; Use global variables only when it is necessary; → Modular, logical and easy to be maintained;
    - Proper indentations, blank lines, to make the block structure more readable and neat
    - Proper in-line comments and instructions and put descriptions in the beginning of the programs
- Documentation: (refer to /**doc** directory under e.g. **lab01/01/doc**)
    - For External users: Instruction on how to compile, run and concise description on what the program does, its input, output etc; (you may put it in the beginning of your program). It's for general users or operators to run the program.

- For internal users: The more details logic, algorithm of how the program process the data; It's for maintaining by other programmers.

Additional guide:

Assumption:

- the size of each word is <= 100 characters (but the numbers of words may be so large and varies)

You may make reference to the functions in the **lab03** and **lab04** to develop the suggested functions in the "assign01 file" (i.e. listcreate(), listAdd(), listserach(), listprint(), listdelete() and other functions you think appropriate).

One of the example of the algorithm in the *main();* filename: "iCountApp.exe"

Main program

- step 1: Read data from "data.txt" word by word
- step 2: Build the linked list and increment the word frequency counts
    - build the linked list word by word:
        - if meet new word → append the word to the end of the list

          else (i.e. find existing word in the list): increment the counter;

        Until the end of the file is reached;
- step 3: Sort the linked list of unique words by counter:
    - Use insertion sort or other sorting methods; (need to specify it in the internal documentation)
- Step 4: Print out the list of sorted unique words and counts

To score high marks

- Add error handling, exception handling; e.g. capture memory allocation failure during malloc() and print out proper message;