**Data Structure Assignment 1**

**Word Frequency Counter**

Deadline: 2012.11.12, 23.59

1. Description

Calculate the frequency of every word appering in the article (not including punctuation marks, space and special characters). Output the k most frequently occuring words along with each frequency and execution time.

1. DO NOT use STL. Implement the entire program by yourself.
2. The algorithm is not limited but must be implemented with “array.”
3. Using C/C++
4. Testing platform is GNU/Linux, gcc version 4.3.2
5. Tip:  You can use time.h ~~function~~ or unix command to calculate the executiom time.
6. Sorting is not required in this assignment. If required, you can use any kind of algorithm to sort. (Hint : It may not be faster to use sorting.)
7. The “words ” are composed of English alphabet and numbers.
8. Input / Output
9. Input (File)

An article. Each word is separated by space, punctuation marks or new line.

1. Execution Process

./executable\_file test\_file k

1. Output (Standard output)

Output k lines of the k most frequently occuring words along with each frequency in order. **Ex: (the,4).** Every pair is separated by new line. And output the execution time.

Attention:

1. Capital letter and small letter are the same.

Ex: “The” and “the” are the same word.

1. The same frequency is considered the same rank. Must output the k most frequency occuring words.

~~A skunk sat on a stump. The skunk though the stump stunk; the stump thought the skunk stunk.~~

Example 1

|  |
| --- |
| **Input file:** |
| A skunk sat on a stump. The skunk though the stump stunk; the stump thought the skunk stunk. |
| **Execution:** |
| ./9955815\_hw1 test24\_hw1.txt 2 |
| **Output:** |
| (the,4)  (skunk,3)  (stump,3)  0.002ms |

Example 2

|  |
| --- |
| **Input file:** |
| I wish to wish the wish you wish to wish, but if you wish the wish the witch wishes, I won't wish the wish you wish to wish. |
| **Execution:** |
| ./9955815\_hw1 test36\_hw1.txt 3 |
| **Output:** |
| (wish,11)  (the,4)  (to,3)  (you,3)  0.003ms |

1. Testfile example

Download “test1\_hw1.txt”from E3.

Note：This file is encoded with UTF-8. Please check out the encoding method if the punctuation is garbled.

1. Requirements

Program

1. You need to turn in the code, test file downloaded from e3 and executable file.
2. Name your executable file “StudenyID\_hw1.” (Ex. 9955815\_hw1)
3. Run your program with parameters (Ex: ./9955815\_hw1 test1\_hw1.txt 3)
4. Using standard output to print out your results and record them in your report.
5. Your program must be readable (Ex. Comments, variable names, function names)
6. Please make sure the test file downloaded from e3 is executable.

Report (Name the file”StudentID\_hw1.pdf”, **Ex: 9955815\_hw1.pdf**)

1. Descript your implementation. (Ex: algorithm, program executing process)
2. Results: test file form E3, execution result and execution time.Please explain the results.
3. Others(Ex. Searching)
4. No more than 2 pages.
5. Grading policy
6. Programing execution (60%)
7. Correctness (30%)
8. Execution speed (30%)

* Top 1~10: get 30%
* Top 11~20: get 27%
* Top 21~30: get 24%
* Etc.

1. Self-test(20%)

Run the testfile from e3 and record the result and execution time in your report.

1. Report (20%)
2. Bonus (10%)

TA will supply a special testfile and you can get the bonus score if your result is cerrect.

1. Submit (e3 will be closed on time)

Compress all your files (including your code, test file downloaded from e3, executable file and report.) Name your compressed file “studentID\_hw1.rar” or “studentID\_hw1.zip”. Upload your compressed file to e3.

Deadline: 2012.11.12, 23.59

No late upload.

1. Attention

NO plagiarize.

If your file name is false, you will get zero.