Data Structure HW1

1. Description

Calculate the frequency of the keyword, not include substring.

Find the position of the keyword in the text, including substring

2. Input / Output

I. Input (From file)

An article in extension of txt, which means the nonsense document. Each word in article is separated by space, punctuation marks or new line. The first word is the keyword. The document's characters will not greater than 1,000,000.

II. Execution Process(Standard input)

Input_file_name.txt Output_file_name.txt

There will be only one line each test. The line indicates the input filename and output filename of the article, separated by a space. Your code should be able to use them for accessing content of article.

see Example 1~3 below in detail.

III. Output(File output)

Output the frequency of the keyword and positions. Notice that the position of the word which contains the answer is begins from 1. Also, if a word contains the keyword more than once, show the number of the word which times it contains.

See Example 1~3 below in detail.

Example 1.

Input file:
at train station the cat at ate a bat, while another cat at the corner attacked by a bat.
Execution:
test1.txt test1out.txt
Output file:
3 13567912131619

Example 2

Input file:
can you can a can as a canner can can a can? cancancan
Execution:
test2.txt test2out.txt
Output:
6 135891012131313

Example 3

Input file:
aa aaa aaasaaaa aa
Execution:
Test3.txt test3out.txt
Output:
2 122333334

Requirements

Program

I. You need to turn in the code.

II. Name your code file "hw1_StudenyID.c/cpp." (Ex. hw1_0416000.c/cpp)

III. Using File output to print out your results to output file.

IV. Your program must be readable (Ex. Comments, variable names, function names)

V. Do not use STL.

Report (Name the file"hw1_StudentID.pdf", **Ex: hw1_0416000.pdf**)

I. Describe your implementation. (Ex: algorithm, program executing process)

II. No more than 2 pages.

Grading policy

I. Sample testing (60%)

Your code can run without crashing and giving correct results to the sample files.

II. Implementation describing (20%)

III. Additional testing (20%)

Your code can run without crashing and giving correct results to the additional files.

IV. Bonus (10%)

using failure function to match string. Also mention in the report.

Submit (e3 will be closed on time)

Compress all your files (including your code(.c/.cpp) and report(.pdf)) Name your compressed file "hw1_studentID.rar" or "hw1_studentID.zip". Upload your compressed file to e3.

Deadline: 2016.10.30 23:59
No late upload.