Minor Assignment 3

ENGINEER 1D04
Dr. William Farmer and Dr. Spencer Smith
McMaster University, Winter 2012

Revised: June 20, 2013

Please use AutoMarker (`automarker.mcmaster.ca`) and Avenue to acquire, test, package and submit your assignment. The procedure for submitting assignments is summarized on Avenue, with additional details provided by AutoMarker. **Please frequently back up your work by creating a submission package in AutoMarker**. This will provide a chance to recover your work in the event of an equipment failure.

Background

`wc` (for "word count") is a well-known unix program that computes the number of lines, words, and characters in a text file. A *line* is a string of characters delimited by newline characters, and a *word* is a string of characters delimited by space characters or newline characters.

Overview

The purpose of this assignment is to write a Python version of the unix `wc` program. Design, implement, and test a program that satisfies the requirements below.

Design, implement, and test a program that satisfies the requirements below.

****Important**:** This assignment will be run through an automated testing program to be graded. The program will make the assumption that any output containing an equals sign (=) is an answer to be marked. **Do not** print output containing the equals sign, except where specified in the requirements below. Outputs must also be printed on separate lines. Additionally, input and output statements must be **exactly** in the order specified. Failure to precisely follow the requirements below will result in a significant loss of marks.

Requirements

1. The program asks the user to enter the name of a text file (f).
2. The program computes:
   a. x = the number of lines in f.
   b. y = the number of words in f.

c. z = the number of characters in f. Do not count newline characters as characters.

3. The program writes x, y, and z on separate lines in a file named `wc_output.txt`.

4. The program is written in Python in a module, NOT in the Python Shell. To create a new module in IDLE, go to File → New Window. You must save this file with a .py extension. For more information on submitting your program, click the "AutoMarker Instructions" button above.

5. Your name, MacID, student number, and the date are given in comments at the top of your Python (.py) file before your program.

6. Your answers to the design questions and the test plan (see below) are given in comments at the bottom of your Python (.py) file after your program.

7. Your program MUST have valid Python syntax and it must run without errors. Ensure that your program runs properly by running it before you submit.

8. You must sign out with a TA or IAI after you have submitted your lab at the submission station. Failure to do so could result in a zero.

Design and Implementation Instructions

1. You may want to use some of the functions (e.g., `split`) from the Python `string` library (see Table 4.2 on p. 96 of the textbook).

2. You should create a sample text file to test your program.

Design Question

What is the best way to compute the number of lines, words, and characters in a file, by reading the file once or reading it three times?

Test Plan

1. Is it necessary that most of the test cases for your program are files having a large number of entries? Explain your answer.

2. Copy and paste the lines from your sample text file to a new document in Microsoft Word. Perform a Word Count in Microsoft Word by going to:

   a. ETB lab: Review → Word Count

   b. JHE lab: Tools → Word Count

Compare the number of characters (with spaces) counted by Microsoft Word to the number of characters obtained with your program. Do these values match? If not, explain why they are different.

Preparation for Major 02

Copy your program under a new name and modify it so that it counts the number of occurrences of a substring s in the input file f. After asking for the filename, your program should ask the user for the substring s. The program will then output the number of times s occurs in f.