CSCI203 ASSIGNMENT ONE

Due 23:30 Saturday 17/08/2024

Must be submitted via Moodle

Task:

Propose an algorithm and implement it with your selected programming language (python, java, or C++). The program shall read and processes a text file, and generate a statistic report on its content.

Your program should:

1. Read the name of a text file from the console.
2. Read in the text file, not all at once. (This can be line by line, word by word or character by character.)
3. Convert the textual content to a sequence of words, discarding punctuation.
4. Convert all letters into the lower case.
5. Store a count (the number of occurrences) of each different word.
6. Sort the words based on the decreasing order of the word counts. If there are multiple words with the same count, sort them alphabetically. (This ordering may be achieved as the words are read in, partially as the words are read or at the end of all input processing.)
7. Output the first ten words in the sorted list, along with their counts.
8. Output the last ten words in the sorted list, along with their counts.
9. Output all 'unique words' (the word count is 1).

Implementation Requirement:

You must choose **appropriate** data structures and algorithms to accomplish this task. *Note that*

1) In the context of this assignment, appropriate choices will be efficient and will not use excessive instructions or data.

2) Where a punctuation mark appears between two letters, the sequence is to be treated as a single word. Thus, '*it's*' will become '*its*', '*you'll*' will become '*youll*' and '*loop-hole*' will become '*loophole*'.

3) You can assume that the input file contains no more than 50,000 different words.

4) Two sample input files "`sample-short.txt`" and "`sample-long.txt`" is provided for you to test your program and produce the program report.

5) you may use any data structures or algorithms that have been presented in class up to the end of week 4. If you use other data structures or algorithms, appropriate references must be provided.

6) Programs must be compiled and executed. Otherwise, a zero mark will be applied.

7) Programs should be appropriately documented with comments.

8) All coding must be your own work.

9) Standard libraries of data structures and algorithms such as STL should **NOT** be used.

10) String class should **NOT** be used, you must define your own string pool.

11) Code from textbooks, the internet, etc may also not be used. Otherwise, you will receive a zero mark.

Report:

A pdf file describing your solution and program output should be produced. This file should contain:

1. A high-level description (in pseudo code) of the overall solution strategy.
2. A complexity analysis of your solution with big-O notation and sufficient justification.
3. A list of all of the data structures used, and the reasons for use them.
4. A snapshot of the compilation and the execution of your program on the provided "`sample-long.txt`" file.
5. The output produced by your program on the provided "`sample-long.txt`" file.
6. The report pdf file should be called `<student number>-a1.pdf`

Submission:

- Please submit your source code and the pdf report as a zip file (named as `<student number used language>-a1.zip`) to CSCI203 Moodle site (Assignment 1 submission folder) before the deadline.
- Please note, the email submission will **NOT** be accepted.
- If an extension (maximally 1 week) is required, please submit an academic consideration via SOLS **before** the deadline.
- Late submission will receive 25% penalty of the assessment weight per day and a zero mark after 3 days.

Marking Guide:

- Programs submitted must work (can be compiled and executed)! A program which fails to compile or run will receive a zero mark.
- If your program produces different output from what is reported in the pdf file, a mark of zero will also be graded.
- A program which produces the correct output, no matter how inefficient the code, will receive a minimum of 50% of the program component of the mark.
- Additional marks beyond this will be awarded for the appropriateness, i.e. efficiency for this problem, of the algorithms and data structures you use.
- Programs which lack clarity, both in code and comments, will lose marks. The total mark will be determined based on both your code and the accompanying design pdf document.