Project 1: Word, Number, and Character Usage Statistics

Due on 02/01/2018 (100 Points)

Educational Objectives: Refresh C/C++ programming skills. Use C++ I/O streams, string class, and STL containers and algorithms. Use makefile to organize and compile programs. Use debugger to identify and address programming problems.

Statement of Work: Implement a program that collects the statistics of word, number, and character usage in a file (redirected as the standard input).

Requirements:

1. Create a subdirectory called proj1.

2. For this project you need to create at least two files: proj1.cpp, and makefile. Both files should be placed in the proj1 directory.

3. The file proj1.cpp should contain the main function, int main(). In the main() function, the program should read the input until it reaches the end, counting the number of times each word, number, and character is used. A word is defined as a sequence of letters ('a'..'z' or 'A'..'Z'). Words are case insensitive ("AA", "Aa", "aA", and "aa" are the same). A number is defined as a sequence of digits ('0'..'9'). Note that both words and numbers can be of length of 1, that is, contain one letter or one digit, respectively. Different sequences represent different numbers. For example, number "001" is different from number "1". Words are separated by numbers or other non-letter and non-digit characters. Numbers are separated by words or other non-letter and non-digit characters. Your program should record the number of times each word, number, and character happens. The program should then output the ten most used characters, the ten most used numbers, and the ten most used words as well as the number of times these characters/numbers/words are used. Since words are case insensitive, the program only outputs lower case words. The characters, numbers and words should be outputted in the descending order based on the number of times they are used. When two characters happen in the same number of times, the character with a smaller ASCII value should be considered as being used more frequently. When two words (numbers) happen in the same number of times, the word (number) that occurs earlier in the input should be considered as being used more frequently.

4. An example executable code of the program proj1.x is provided to you. In proj1.x, the output related to the display of characters, words, and numbers is aligned in the following manner: the width of the column of the characters, words, and numbers is the length of the longest words and numbers to be displayed, plus five (5). You should make the outputs of your program the same as those of 'proj1.x'. When printing characters, use '\t' for tab and '\n' for newline. All other characters should be outputted normally.

5. Write a makefile for your project that compiles an executable called proj1.x

6. You are encouraged to use any C++ STL containers and algorithms. You should also use C++ string class instead of the built-in string type.

7. Your program must be able to compile and run on linprog.

Example executable code

Click here to download the example executable code and 4 test cases. The executable code was compiled on a linprog machine. One bonus point is given to the first student who identifies a problem in the example executable code (no known problems with the provided code).

You need to redirect one of the test case files as the standard input to the executable code, for example:

proj1.x < test0

Submission

Tar all the source codes, including proj1.cpp and the makefile into a single file and submit online via Canvas (don't use dropbox; use the link following the assignment to submit). Make sure you tar your programs correctly. You are responsible for incorrect submissions (for example, empty tar file). You can download your submission and untar the submission **under a different directory** to make sure that you do include all the right source files. Same late policy applies if the submission is incorrect and you need to submit a new version.

You should be able to download your submission and verify if it is correct. If you cannot download the submission, contact the Canvas support team at FSU.

Note that in addition to the provided test cases, we will also test your program using additional test files. Your program must be able to pass all the test cases in order to obtain a full score for the corresponding components.