Due Date
Wednesday, April 16, 2014

Delivery method
Deliver a single archived file named pa5.tar (all lowercase) as an **attachment** to an email that you send to the class account csc3320@orion.ucdenver.edu. Put PA5 in the subject field, and your name in the body of the email. As usual, use lowercase for file names and appropriate names such as pa5.cpp pa5functions.h, etc.

Program objectives
The objectives of this assignment are as follows.
An ability to use current techniques, skills, and tools necessary for computing practice (ABET i).

Value
This program is worth 15 points. The distribution of points will be as follows.

Criterion	Value
Global functions	1
3320 list	1
STL map	2
Program style	3
Correct output with annotation	6
STL istream_iterator	1
General code usage	1

Problem
Part I
Read the name of a line-oriented text file from the command line in argv[1] and create a concordance. The concordance will consist of a list of the words that appear in the file, along with the line numbers on which they appear. Write the concordance to a file whose name you read from the command line in argv[2].

Input
A text file with space delimited words (no contractions or hyphens) with normal punctuation. The program should read the name of the input file from argv[1]. Read the input using **std::istream_iterator<T>**.

Output
A file containing a formatted concordance of the input file. The name of the output file to create will be available in argv[2]. Each line of the file should contain a **unique** word from the input file, along with the comma delimited line numbers on which it appears. The words should be converted to lowercase and sorted lexicographically. Here is an example of the output. Notice the formatting—no ragged columns.

WORD LINES
although 2, 7, 9

brought 1, 3
…..

The first column should have a field width equal to size_of_largest_word + 2, where size_of_largest_word refers to the maximum number of characters in any word in the file. The first column should also be left justified.

Part II
From the file you read in Part I, compute the frequency-of-occurrence of each letter of the alphabet (case does not matter).

Input
The text file from Part I read with std::istream_iterator<T>.

Output
1. A file containing each alpha character that appeared in the file along with the number of its occurrences. The file should be formatted as follows.
   a. 2 rows, 13 columns, listing the alpha characters a-m on row 1 and the corresponding frequencies on row 2.
   b. 2 blank lines.
   c. 2 rows, 13 columns, listing the alpha characters n-z on row 1 and the corresponding frequencies on row 2.
   d. Columns should be evenly space (no ragged columns) with left justification.
2. A formatted, vertical star (*) plot of the frequencies of the letters used in the file in the order from most frequent to less frequent. Here is a portion of a typical plot for the top 4 letters of the alphabet for some string. Your plot must show all 26 letters and must appear on the standard output.

```
*
*
*
*
*   *
*   *   *
*   *   *   *
*   *   *   *
*   *   *   *
e   d   t   n
```

Minimum program requirements
1. Greeting with pause (greeting can explain the whole program).
2. Read the input file from argv[1], open it, and read the data using std::istream_iterator<T>.
3. Create a concordance and write it to an output file named in argv[2].
4. Create the letter frequencies and write to an output file named in argv[3].

Class requirements

1. There should be at least letter_frequenices and concordance classes.
2. Use the std::map<K,V> (one or more). There should be at least one map with key=std::string and value type=3320 linked-list.

Notes

1. Be sure to use error checking where appropriate.
2. You may use other STL components if you deem necessary.