# CPT 244 Program 4

Dusty Book Researchers wants a program that will help then do some analysis on books that they have. Yes, a database and some good SQL queries would be a better approach, but they are cheap and so are we so they've contracted our services.

You can assume that the books are all ASCII text with words separated by whitespace. For the purposes of this class, all punctuation has been removed. In addition, all the letters are lower case. This eliminates the need to fix the case in your program. The data file that you are given is simply a bunch of lines of text. Occasionally there are misspelled words. You do not need to be concerned about that as they will simply show up as another word in your list.

The data should be stored in a Dictionary. Do not copy all the data to another structure. However, you may copy the data to be output to an alternative data structure that is sortable, since that may make things considerably easier.

Use ReadLine and Split when reading in the file. There are alternatives, but these can sometimes give you odd results due to the linux/windows differences in how newline is stored.

You are given some leeway in how you use your Dictionary. You may use the .NET methods that are found in the Dictionary class. However, do NOT use LINQ. (If you have no idea what LINQ is, google C# and linq.) This program is to help you see some of the uses of Maps as well as where Maps are less desirable as data structures. Using LINQ hides the complexity and makes the program trivial. (If you want to have a second solution that uses LINQ, feel free. Place those methods in a separate project so that there will be no ***using*** statement for linq in the file that has your actual work.)

## For a grade of ‘C’:

Read all of the book into a Dictionary, counting each occurrence as they are read in. Print the following:

* The total number of words in the book
* The number of different words in the book; i.e. how many entries are in your Dictionary.
* The word that occurs the most number of times, and how many times it occurs.
* The number of times ‘*count’* occurs.
* The number of times ‘*may’* occurs.
* The number of times ‘*color’* occurs.

## For a grade of ‘B’:

After printing the above information, provide a while loop that presents the user with a menu that allows them to select the following. After the user makes the selection, prompt for the word or number needed to show the answer.

1. How many times a particular word *S* appears, where *S* is input by the user.
2. All the words that appear *X* times; where *X* is input by the user.
3. All the words whose length is equal to *X*, again where *X* is input by the user; both the **word** and the **number of appearances** should be printed.

## For a grade of ‘A’:

* In the initial output, display the **ten** most used words.
* Sort the answers for ***‘b’*** above *alphabetically* by the words.
* Sort the answers for ***‘c’*** above *numerically* by the number of appearances, highest to lowest.

## **Questions:**

Some questions to answer when you are finished. Submit your answers in a separate document.

1. Using a Dictionary, how long does it take to do each of the following (i.e. what is the complexity. Assume that there are N words in the file and K words in the Dictionary.

* Reading in the file
* Getting the highest word count
* Getting the average word count
* Getting the number of times a specific word is used
* Listing the words that have a count of 25
* Listing the words and count that have a length of 7
  + finding the words
  + sorting the words

If allowed to use any data structure, would you change the data structure to be more efficient? If so, what structure would you use and why? If your data structure is sortable, on what field would you sort the data? Why? How would your new data structure affect the answers in the question above?