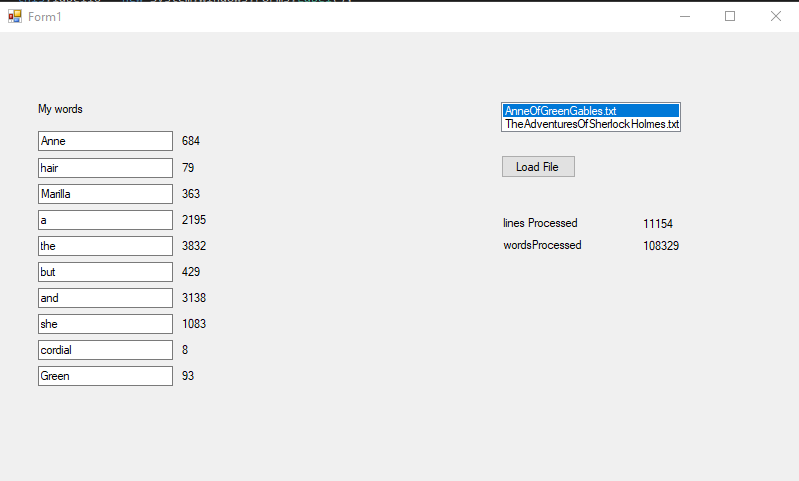
# Lab Exercise #3: Strings, Files, and Loops

Assignment #1 is on your plate, so this lab is intended to be a relatively easy one.

You are provided with a Lab 3 Exercise Starter Package: Word Frequency. It will read in a file line by line, split each line into words, and store how many times each word appears.



For each word that you type in your word list, it will tell you how many times that word appears in the file.

The basic word frequency list is working, but has some issues, and you can help solve some of them.

## Core Task

Two small tasks to change the functionality of our word frequency program.

### Frequencies

We have counts, but not frequencies. I want to see some percentages! Change the program so that the label contains the frequency as well as the count.

That is, for the example above, beside ‘and’ we should see “3138 (2.90%)”.

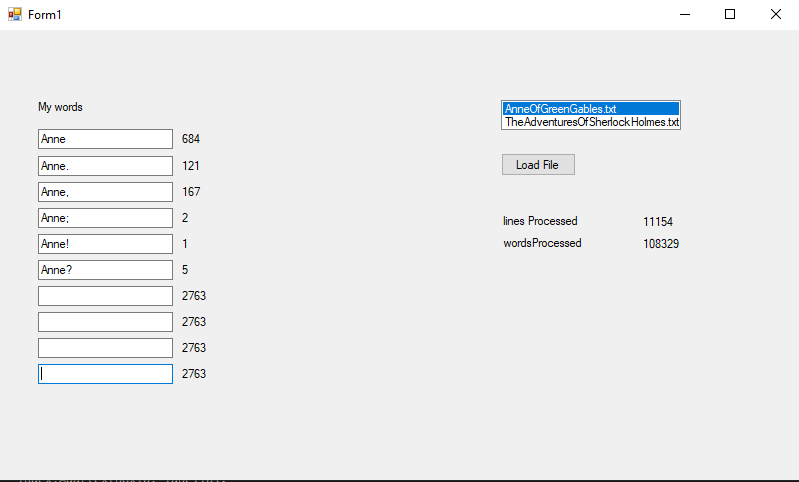
Hint: the frequency is the count of the word divided by all the words processed, both of which are already calculated for you.

Hint: You can get the percentage to two decimal places. You can multiply the double by 100, then round, then get the number into the string and add the percent sign yourself. Or, C# has String.Format, which lets you do stuff like format a number like ‘Currency’ or ‘Percent’ or ‘Scientific notation’, etc., without you having to do all the work.

Hint: Format specifiers can be found here:  
  
<https://learn.microsoft.com/en-us/dotnet/standard/base-types/standard-numeric-format-strings>

### Punctuation!

The function I used to split lines on spaces isn’t very smart, as you can see below:



Fix this problem so that all the above words get stored as ‘Anne’ – that is, remove the punctuation in the screen shot above.

Hint: There is a String.Replace method that can take two strings, the string to replace and the string to replace it with. E.g. “Scott”.Replace(“tt”,”ph”) returns “Scoph”. Super-hint: the second string can be empty, and the first string can be just one character.

Completing these two parts of the Core Task is worth 1.5 / 2 for the Lab Exercise.

## Extensions

Here are some fun extensions to the core task. You must complete and submit at least one of them to get full marks on the Lab Exercise. I encourage you to either try or at least think about more than one of them.

* Do a better job on the punctuation removal.
  + Hint: there is a Char.IsLetter() method that tells you if a character is a letter
  + Hint: since a string is a collection of characters, you can iterate through them
    - foreach(char c in word) { // do something }
  + Hint: the // do something can be to add the character ‘c’ to the new word that you are building, but only if it is a letter
* Write the entire word list with frequency counts to an output file.
  + You probably want to create an “Output” directory beside input.
  + If you do that, your filename will be something like “../../Output/output.txt”
  + Hint: you probably want to create a StreamWriter
    - E.g. StreamWriter writer = new StreamWriter(filename);
    - And then foreach entry in your Dictionary wordCounts, build a string and do writer.WriteLine
* When you’ve built your dictionary, find the most common word and show it in the form.
  + Hint: foreach word in the dictionary, look at its count, and see if it is bigger than the ‘biggestSoFarCount’, and if so, set a new ‘biggestSoFarWord’ and a new ‘biggestSoFarCount’
  + Hint: Or sort the collection by value, but that is a bit tricky with a Dictionary, it doesn’t come built-in.
* Compare the counts between the two files
  + That is, show how many times the word appears in Anne \*and\* how many times it appears in Sherlock.
    - Hint: Change the load file button to always load both files
    - Hint: Build a different dictionary for each file.
* Build a character count dictionary and display counts on screen
  + You’ll need another dictionary for the characters
  + Hint: Foreach(char c in word) { // processCharacter }

## Submissions

Please submit the Form1.cs file and a screenshot showing your form solving the issues and displaying the properly formatted counts and outputs.