CSE101 – Spring 2020
Programming Assignment #6
(12 points, Submission due date: 19 May 2020)

Instructions

For each of the following problems, create an error-free Python program.

- Each program should be submitted in a separate Python file that follows a particular naming convention: Submit the answer for problem 1 as "Assign6Answer1.py" and for problem 2 as "Assign6Answer2.py" and so on.
- These programs should execute properly in PyCharm using the setup we created in lab.
- At the top of every file add your name and Stony Brook email address in a comment.
- Write 2 test cases to test your code for all problems that do not take user input.
- Download these supplemental files – unzip the files and add them to your project directory. The problems will refer to specific files that you will load data from.

Problems

Problem 1: Most common characters (with dictionary) (3 points)

Below is a prior quiz problem. Solve this problem again, but this time use a dictionary to store the counts of each letter. Your solution **must use a dictionary in order to receive any points** – you cannot just submit a prior solution that did not use dictionaries.

Write a function that will take a string as a parameter and prints on one line the top three most common characters in the string, along with how many times each character occurs. The most common character should be printed first, followed by the second most common character, followed by the third most common. If the occurrence count is the same, any character can be used (e.g. for "Google" could use G,O,E or G,O,L).

You should make the string uppercase using the `upper()` method, so that it will count upper and lowercase letters together (e.g. "Google" has 2 "G" characters when it is made uppercase). You can assume the string always has at least three distinct characters. Some example test cases:

```
>>>threeMostCommonCharacters("Google")

G:2, O:2, E:1

>>>threeMostCommonCharacters("Mississipi")

S:4, I:4, P:1

>>>threeMostCommonCharacters("Daimlerchrysler")

R:3, E:2, L:2
```

Problem 2: Iris dataset (4 points)

Refer to the file `irisdata.txt`. This data set is commonly used in machine learning experiments and consists of 50 samples from each of three species of Iris flower (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

Write a Python program to read the data from the text file and print the averages of the sepal length, sepal width, petal length, and pedal width for all three types of Iris flowers.

Expected Output:

```
Class Iris-setosa:
Average sepallength: 5.006
Average sepalwidth: 3.418
Average petallength: 1.464
Average petalwidth: 0.244

Class Iris-versicolor:
Average sepallength: 5.936
Average sepalwidth: 2.77
Average petallength: 4.26
Average petalwidth: 1.326

Class Iris-virginica:
Average sepallength: 6.588
Average sepalwidth: 2.974
Average petallength: 5.552
Average petalwidth: 2.026
```

Problem 3: Spellchecker (5 points)

Refer to `wordlist.txt` and `speech1.txt` as part of this assignment.

Automated spell-checkers are used to analyze documents and locate words that might be misspelled. These programs work by comparing each word in the document to a large list of words. Any word not found in the list is flagged as potentially incorrect. Write a program to perform spell-checking on a text file. Your program should prompt for a file to analyze and then try to look up every word in the file using binary search. If a word is not found in the dictionary, print it on the screen as potentially incorrect.

Directions to solve the problem:
- Read all the words from `wordlist.txt` and store them in a list.
- Sort the list using the mergesort algorithm.
- Write a recursive binary search algorithm that searches a given word in a dictionary.
- For each word in speech1.txt file use the binary search algorithm to check whether the word is spelled correctly. The case of word does not matter.
- Print a list of misspelled words from the "`speech1.txt`" file.

You can use the example code we have previously provided for implementing the algorithms.

If run correctly, your program should output:

```
All incorrect words are:
['spich', 'draams', 'comfrot', 'lifehacker', 'nataral', 'preditcable',
'becume', 'preson', 'disracted', 'dobbler', 'aspirtions', 'aksed',
'watned', 'atempt', 'dicipline', 'togther', 'chnaged', 'oponed',
'sometmes', 'miror', 'conseersation', 'negtive', 'alwys', 'yuorself',
'confortable', 'everething', 'comfrot']
```