Timed Lab 1 – String Stats

This is a ***practice*** Timed Lab; this Timed Lab is worth 20 **Attendance & Participation** points.

For this Timed Lab, you *may* use	However, you *may not*
• Course notes • Homeworks • Recitation assignments • Other course material • Any material you may find on the Internet that don't involve communicating "live" with other people.	• Communicate with other people/students in real-time via any means. This means no Facebook, email, Piazza, IM, IRC, cell phones, Google Talk, smoke signals, etc. • Share code with other students. • Look at other students work.

The TAs will be available to answer clarifying questions about the problem, but they are not permitted to give assistance with debugging code, program logic, etc. You will have an entire recitation period to work on this assignment; this time begins *exactly* when your recitation begins and ends *exactly* when your recitation ends: No extra time will be given if you arrive late, except in highly extenuating circumstances that must be approved by Dr. Summet.

T-Square will not permit any late submissions; ensure that you submit your code to T-Square several times to prevent earning a zero due to you being unable to submit. Your TAs will give a verbal warning 10 and 5 minutes before the end of the recitation period; you should submit at these times.

In your collaboration statement, if you use code from somewhere that is *not* a class resource (i.e. not listed on the course calendar), please list where this code came from. Ensure that you fill out the header at the top of the file.

Problem Description:

You have been tasked with writing part of a module that will output statistics about a given string. Your first task is to write a function that will return the number of times a particular character appears. Then, you will be asked to find the character(s) that appears the most number of times in the given string and return this character(s) along with the number of times these character(s) appear.

If you wish, you may write helper functions in addition to the two functions that are required; however, you are not required to do so. Below are the descriptions of the functions you should write.

Required Functions:

countChar - This function should take in two parameters: the first is the string you wish to look through, and the second is the character which you are finding the number of in the given string. Note that case does not matter; an uppercase character is the same as a lowercase character. You may assume that the input for the second parameter is always exactly one character. You may have to count white space (tabs/newlines/spaces) or punctuation (commas, periods, etc...), but they do not have an "upper" and "lower" version. The str.lower() function may be useful to you to ignore case.

Test Cases:
countChar(“This is a test”, “t”) → *3*
countChar(“This is a test”, “T”) → *3*
countChar(“This is a test”, “x”) → *0*
countChar(“This is a test”, “i”) → *2*
countChar(“”, “b”) → *0*

maxChars - This function should take in one parameter, the string which you wish to find the maximum-occurring characters in. Your function should return different things depending on the string you are given. If the string you are given is empty, you should return None. If the string you are given only has one character with the maximum count, ("aaabc") you should return a tuple where the first element in the tuple is the character with the maximum count, and the second is the number of times that character occurs. ("a",3)

If the string you are given has more than one character with the maximum count, ("aabb") you should return a tuple which contains two elements. The first element is a tuple that contains all of the characters that occur the most times, and the second element is the number of times that one of the characters with the maximum count occurs. (('a','b'), 2)

As with the countChar function, the case of a character does not matter; for the purposes of this function, and upper and a lower case character are the same character.

Test Cases:
maxChars(“”) → *None*
maxChars(“Testing123”) → *('t', 2)*
maxChars(“This is a test”) → *(('t', ' '), 3)*
maxChars(“aaabbbcccdddeeef”) → (('a', 'b', 'c', 'd', 'e'), 3)

Grading:

countChar		**6**
Function header correct	1	
Returns count of character in string	1	
Ignores case of characters	2	
Gets correct count of character in string	2	
maxChars		**14**
Function header correct	1	
Returns None if argument is empty string	1	
Correctly finds max character if only one max	3	
Correct tuple returned if only one max	2	
Considers possibility of multiple max chars	1	
Correctly finds all max characters if more than one	4	
Correct tuple returned if more than one max	2	