Timed Lab 1 – Library Helper

This is a Timed Lab; this Timed Lab is worth 45 **Attendance and 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 or the Head TA.

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.

Note that you must check out with your TA before you leave the recitation room. If you do not check out with your TA or you modify your submission after you leave the recitation room, **you will receive a grade of zero on the timed lab.** No submissions will be accepted after T-Square closes the assignment (i.e. it will not let you submit).

Problem Description:

You have been contacted by a library system to assist them with library operations. Specifically, they need help with getting the titles for their books into the correct format, determining which patrons have too many books checked out to them, and computing a fine for patrons that have too many books checked out at any one time.

NameFormatter:

The purpose of the NameFormatter function is to convert titles to the correct format. Titles of books that start with the words “A”, “An’, or “The” need to be modified such that the “A”, “An”, or “The” is moved to the end, with a comma between the title and the “A”, “An”, or “The”. For example, “The Catcher in the Rye” should become “Catcher in the Rye, The”, or “A Tale of Two Cities” should become “Tale of Two Cities, A”. Any other title should be left alone.

For this task, you should write a Python function named NameFormatter that takes in one parameter, the title you wish to format, and should return the properly formatted title. This function should also be able to handle titles that do not start with “A”, “An”, or “The”.

BookHogFinder

This function named BookHogFinder should take in two parameters. The first is a list of patron names and the number of books they have checked out, and the second is the cutoff for reporting a patron who has too many books checked out at once. The list of patrons and the number of books they have checked out was formatted strangely by the U[sic]GA grad who tried to write this system before you, and is formatted as follows: ["Patron 1", 3, "Patron 2", 15, "Patron 3", 4, …]. You quickly note that it is an alternating pattern of patron name and the number of books checked out to that patron directly after (i.e. Patron 1 has 3 books, Patron 2 has 15 books, Patron 3 has 4 books, and so on). After processing the entire list, you should return a list of all the patron names who are **not** under the cutoff. Make sure that you do not modify the original list!

FeeGenerator

Your job is to calculate fees for the book hogs, dependent upon the number of books they have checked out. Your function, named FeeGenerator, should ask the user to input two pieces of data: The name of the patron and the number of books the user has checked out. For the first 3 books checked out, there is no fee. For the 4th to 9th book checked out, there is a $1.25 fee per book. For any book after the 9th book, the fee is $0.755. When you have calculated the amount, you should print the message "The fee for patron <PATRON_NAME> is $<FEE_AMOUNT>" without the quotes or <>, and with PATRON_NAME and FEE_AMOUNT replaced with their respective values, with the fee amount displayed to two decimal places. This function should not return a value.

Grading:

NameFormatter

+2 – Correct function header
+2 – Correct return type
+3 – Handles cases with "A" correctly
+3 – Handles cases with "An" correctly
+3 – Handles cases with "The" correctly
+2 – Handles cases without "A", "An", or "The" as the first word correctly.

BookHogFinder

+2 – Correct function header
+2 – Correct return type
+3 – Correctly identifies and implements list pattern.
+3 – Handles any number of patrons in the list.
+3 – Identifies and saves only patrons at or over the cutoff.
+1 – Does not modify the original list
+1 – Returns the correct list.

FeeGenerator

+1 – Correct function header
+1 – No return value
+1 – Asks for user input
+1 – Prints result.
+2 – Handles 0-3 book case correctly.
+3 – Handles 4-9 book case correctly.
+3 – Handles 10+ book case correctly.
+3 – Prints to two decimal places