COMP 1012 Fall 2017 Assignment 2

Due Date: Friday, July 13, 2018, 11:59 PM

Material Covered

- User defined functions
- return

Notes:

- Name your script file as follows: <LastName><FirstName>A2Q1.py. For example, LiJaneA2Q1.py is a valid name for a student named Jane Li. If you wish to add a version number to your file, you may add it to the end of the file name. For example, SmithRobA2Q1V2.py is a valid name for Rob Smith.

  Spyder automatically adds the .py part (called the extension) to your file name, so you don't have to type that part when saving the file. However, please ensure that your file name DOES include the .py extension, since our marking software looks for it.
- Name your output file as follows: <LastName><FirstName>A2Q1output.txt. For example, LiJaneA2Q1output.txt is a valid name for a student named Jane Li.

  In order to prevent Spyder from automatically adding the .py extension to your output file name, you must type the complete file name, including the .txt extension, when saving your output file.
- Follow the posted programming standards to avoid losing marks. Check your script for adherence to the programming standards by using CheckStandardsV2.py, which will show you exactly where your script does not comply. We will use the same script to assign marks for your assignment, so you have NO EXCUSE for losing these marks.
- You must complete the ***Blanket Honesty Declaration*** checklist in order to submit your assignment. This applies to all assignments in COMP 1012.
- To submit the assignment follow the instructions on the course website carefully. You will upload both script and output files, via the course website. We will demonstrate the assignment hand-in procedure in lectures. There will be a period of about a week before the due date when you can submit your assignment. ***Do not be late!*** If you try to submit your assignment after the late submission deadline, you will get a message indicating that the deadline has passed.

Question 1—Statistical analysis of large datasets [50 marks]

Description

The description for this problem is exactly same as the assignment 2 question 1. But here you need to use functions for reading the file, finding standard deviation, squared difference, max and mean value.

What to do

Create a module with name StatisticsCalculation. The statistics calculation module contains following functions:

`calculateMean(data):`

The parameter data is a list of floating point values. Calculate and return the mean.

`calculateStdDev(data, mean):`

Given a list of data and the mean of the data, calculate and return the standard deviation.

`findMin(data, mean):`

Given a list of data, it will return the min.

`findMax(data, mean):`

Given a list of data, it will return the max.

`findPassedFailed(data, cutoffMark):`

Given a list of data and a cutoff mark after which students will be considered failed, it will return a tuple telling number of passed and failed students.

Create another module with name InputOutputHandler. This module contains the following functions:

`printCSVResults(headings, means, stdDevs, mins, maxs,passed, failed):`

The parameters headings, means, stdDevs, mins and maxs are lists containing the column headers and statistics from the csv file. passed and failed are two optional parameters with default value set to 0. You need to print the statistics from this function.

`readCSVFile(fileName):`

The parameter fileName is a string. Open the file and read the header row and data from the csv file. Return the header and data. This should be able to read any number of columns from the csv file.

Create a main function. And from the main function call these functions to find the statistics. You may call the functions in any order but the result should be exactly similar to assignment 2 question 1.

```
Enter file name: Grades.txt
 Column Names |     Mean      | Std Deviation | Highest Score | Lowest Score
--------------+---------------+---------------+---------------+---------------
       Midterm|          22.25|          10.49|          40.00|           1.00
--------------+---------------+---------------+---------------+---------------
         Final|          39.12|           9.18|          60.00|           1.00
--------------+---------------+---------------+---------------+---------------

Total Number of students: 100
Passed in the exam: 81
Failed in the exam: 19


Programmed By Instructor: Sheikh Jubair
University of Manitoba
Computer Science
```

Question 2—Guessing Game [50 marks]

Description

An Armstrong number is a number such that the sum of its digits raised to the third power is equal to the number itself. For example, 371 is an Armstrong number, since 3**3 + 7**3 + 1**3 = 371. In this program, you first need to find all the Armstrong numbers between 0 to 1000. Among those numbers, computer will randomly pick an Armstrong number. After picking a random Armstrong number by computer, you need to guess that number. So the computer will ask you to enter a number. If your guessed number does not match with the Armstrong number computer selected, the program will give you a hint about the range in which the Armstrong number lies and will ask you to enter a new number. The program will continue till you make the correct guess. You also need to print how many attempts you need to correctly guess the number.

What to do:

Create following functions:

`findArmStrongNumber(num):`

This function will take a number as a parameter input and return true if the number is Armstrong number. Otherwise it will return false.

`selectArmstrong(armstrongs):`

This function will take a list of Armstrong numbers as parameter input and return randomly selected one Armstrong number. To do this, you need to import random module. random.choice(armstrongs) will return a randomly selected Armstrong number from the list.

predictArmstrong(selectedArmstrong):

This function will take the randomly selected Armstrong number as parameter. It will continuously ask you to enter a number until your guess is equal to selectedArmstrong number.

main():

In the main function, use a loop to go from 0 to 1000 and pass each number to the findArmStrongNumber function. If the function returns true, add the number to an Armstrong number list. Then pass the Armstrong number list to selectArmstrong function. This function will return an Armstrong number from the list you passed. Then pass the selected number as a parameter to predictArmstrong function.

```
Make a prediction: 900
Your predicte armstrong number is too high

Enter a value between 0 and 900: 10
Your predicte armstrong number is too low

Enter a value between 10 and 900: 50
Your predicte armstrong number is too low

Enter a value between 50 and 900: 600
Your predicte armstrong number is too high

Enter a value between 50 and 600: 100
Your predicte armstrong number is too low

Enter a value between 100 and 600: 500
Your predicte armstrong number is too high

Enter a value between 100 and 500: 200
Your predicte armstrong number is too low

Enter a value between 200 and 500: 300
Your predicte armstrong number is too low

Enter a value between 300 and 500: 400
Your predicte armstrong number is too low

Enter a value between 400 and 500: 407
Your Prediction is correct!!!
You need 9 attempts
```