CSE101 – Spring 2020

Programming Assignment #3

(13 points, Submission due date: 16 April 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 “Assign3Answer1.py” and for problem 2 as “Assign3Answer2.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.

Problems

Problem 1: (3 points)

Given an integer named `area` which represents the area of a rectangle, write a function `minPerimeter` that calculates the minimum possible perimeter of the given rectangle, and prints the perimeter and the side lengths needed to achieve that minimum. The length of the sides of the rectangle are integer numbers.

For example, given the integer `area = 30`, possible perimeters and side-lengths are:

(1, 30), with a perimeter of 62
(2, 15), with a perimeter of 34
(3, 10), with a perimeter of 26
(5, 6), with a perimeter of 22

Thus, your function should print only the last line, since that is the *minimum* perimeter. Some example outputs:

```
>>>minPerimeter(30)

22 where rectangle sides are 5 and 6.

>>>minPerimeter(101)

204 where rectangle sides are 1 and 101.

>>>minPerimeter(4564320)

8552 where rectangle sides are 2056 and 2220.
```

Problem 2: (2 points)

Write a function `fuelEfficiency` that computes the fuel efficiency of a multi-leg journey. The function will first prompt for input for the starting odometer reading and then get information about the series of legs on the journey. For each leg, the user enters the current odometer reading and the amount of fuel used in liters (separated by a comma). The user signals the end of the trip with a blank line. The program should print out the kilometers per liter achieved on each leg (to 1 decimal precision). The program should also print out for the trip the total distance, total fuel consumed, and fuel efficiency for the entire trip. See the example below:

```
>>>fuelEfficiency()

>>>Enter initial odometer reading: 1000

>>>Enter leg info (odometer reading, fuel used): 1040,2

20.0 km per liter

>>>Enter leg info (odometer reading, fuel used): 1080,3

13.3 km per liter

>>>Enter leg info (odometer reading, fuel used): 1190,10

11.0 km per liter

>>>Enter leg info (odometer reading, fuel used): 1250,5

12.0 km per liter

>>>Enter leg info (odometer reading, fuel used):

Total distance traveled = 250 km.

Total Fuel used = 20 liters. Fuel Efficiency = 12.5 km per liter.
```

Problem 3: (5 points)

Create a function `longestSubstring` that takes as input a string parameter and returns the longest substring that contains no *consecutive duplicate* characters as a list. If there are two or more substrings with the same length, then return them all as separate items in the list. If there is only one longest substring, return it as a list of length 1. Here are some examples:

```
>>>longestSubstring('abcbba')

['abcb']

>>>longestSubstring('abcdeffghijkkklmnopqr')

['klmnopqr']

>>>longsub('abcdeffghijkkklm')

['abcdef', 'fghijk']
```

Problem 4: (3 points)

Write a function called `calculateHighAverageLow` that takes a **list of lists** of numbers (a 2-dimensional list) and calculates the highest number, the average, and the lowest number for each list within the list. The function should then return a new 2-dimensional list with the results of those calculations for each list in the format:

[[high1, average1, low1], [high2, average2, low2]]

You can assume every list will have at least one number. You should account for both positive and negative numbers, as seen in the 3rd example below.

Below are example outputs given three different lists of lists:

```
# Note the second list only contains a single value, 5

>>>data0 = [[10, 15], [5]]

>>>calculateHighAverageLow(data0))

[[15, 12.5, 10], [5, 5.0, 5]]


>>>data1 = [[20,30,10,5,5], [36,25,20], [17,2,12,14]]
>>>calculateHighAverageLow(data1)

[[30, 14.0, 5], [36, 27.0, 20], [17, 11.25, 2]]


>>>data2 = [[20,-25,-30,35,-15], [15,10,-5,25,20,17.5]]
>>>calculateHighAverageLow(data2)

[[35, -3.0, -30], [25, 13.75, -5]]
```