CSC241 – Lab 4

Instructions:

1. **Read the HW guidelines** posted in d2l.depaul.edu > admin.

2. Your solution file MUST be named **lab4.py**. In a comment, include the name(s) of any collaborators.

3. To receive **full credit**, the **names** of files, functions and the **output** must be ***exactly*** as indicated here.

4. **Test your code** by downloading the file **lab43TEST.p**y in the same working folder and typing in the shell:

   ```
   >>> import doctest
   >>> doctest.testfile('lab4TEST.py')
   TestResults(failed=0, attempted=23)
   >>>
   ```

   If you prefer, you can make the test run automatically by adding the following lines at the bottom of your hw2 file/modfule. Every time you run the module, the doctest will be performed.

   ```
   if __name__=='__main__':
       import doctest
       print( doctest.testfile('lab4TEST.py'))
   ```

5. **Copy the** results from the above doctest (it may be longer if you have erorrs) and paste at the top of your submission file. Comment the results out using a multiline comment (triple quotes).

Problems:

1. Write a function `lineLengths` that accepts one argument, the name of a file. The function then ***prints*** an expression showing that the sum of the characters in each line sums to the total number of characters in the file. For example, the first line in `test1.txt` has 15 characters , the second has 20, …, the last line has 24, and there are 82 total characters in the file. Note that the final 0 is a little hack as it does not correspond to an actual line and is printed only so that the sum ends with a number rather than with a `'+'`. (How would you do it without the +?) Here are test runs:

```
>>> lineLengths('test1.txt')
15+20+23+24+0=82
>>> lineLengths('test2.txt')
16+21+49+24+0=110
>>>
```

2. Create a function `monogram` that accepts a single argument, a string containing a name (first, last and perhaps many middle names), and returns a monogram, the first initial of each name in capitalized form. For example:

```
>>> monogram('cher')
'C'
>>> monogram('George Washington')
'GW'
>>> monogram('george herbert walker bush')
'GHWB'
>>>
```

3. Write a function `pay` that accepts two numerical arguments, the first an employee's hourly pay rate and the second the number of hours the employee worked. The function computes and **returns** the employee's pay for the pay period. Any hours beyond 40 and less than 60 should be paid overtime at 1.5 times the regular hourly rate. Any hours greater than 60 should be paid at 2 times the hourly rate. The following shows how the `pay` function would be used on several examples:

```
>>> pay(10,35)
350
>>> pay(10,45)
475.0
>>> pay(10,61)
720.0
```

4. Implement a function `average` that takes one string argument, the name of a text file. The function should **return** the average length of a word in the file. You may assume that the file contains no punctuation. If the file is empty, the function should return 0.0. The following shows what the function would return when called on several sample files:

```
>>> average("example1.txt")
4.352941176470588
>>> average("example2.txt")
4.523809523809524
>>> average("empty.txt")
0.0
>>>
```