CS 121 – Lab #7 – File I/O

In today's lab, you will write two programs to practice file I/O. The first program will feature file output and the second one will focus on file input. To begin, create a new folder on your USB drive or account called lab07. On the class Web site, there are some files you will need to download for this lab. They are available on cs.furman.edu/~chealy/cs121/lab07. Open Anaconda Sypder as usual.

Part 1: Printing many primes

Let's write a program that asks the user to enter a large positive number n. The program will then find all of the prime numbers between 1 and n, inclusive. But this could be quite a long list. For example, there are 5133 primes between 1 and 50,000. It would be impractical to print all of these numbers to the screen: the user would not be able to see all of them. It would be wiser to write all of this output to a file.

Your program should have a structure like this:

```
Create an output file primes.txt.
Ask the user for the finishing number.
Set count to zero.
n = 2
while n <= the finishing number,
    See if there are any divisors from 2 to sqrt(n).
    If so, then n is not prime.
    If n is prime, count it and write it to the file.
    n = n + 1
print a message on the screen saying we're done,
and indicate the number of primes found.
```

Make sure that when you print prime numbers to primes.txt, you have one number per line. Test your program with 50,000 as the finishing number, and verify that it found the correct number of primes.

Once you have verified that the output is correct, let's add one more feature. When we enter a large input value, your program could take a long time. How can we tell it's doing useful work? Almost all of the program's time is spent in the outer while loop. So, at the beginning of this loop, it would be a good idea to print some progress information to the screen. The computer operator can see this output and be reassured that the program is working towards a final answer.

At the beginning of your while loop, you can include code to occasionally print which iteration it is currently working on. We want to do something like this:

```
    if n mod 10000 equals 0,
        print n
```

Now, run your program with a larger number like 500,000 or 1,000,000, to observe how the program reports its progress.

Part 2: Computing a centroid

One trend in American history is that the population has been spreading westward. Early in the nation's history, the center of the US population was near Baltimore. But today it is in Missouri. How would we calculate the center of population? The general mathematical problem is computing what is called a centroid, or a weighted average of all of the populated areas of the country.

Let's write a program that computes the centroid of a 2-dimensional region. The .txt files that you downloaded for the lab will be used as input. The file "input.txt" is very short, and is intended to be a simple check of your program's calculations before you get too far along. The other input files are much larger representing real-world input.

Each line of the input file contains a location represented by x and y coordinates (or latitude and longitude), followed by a weight. The x and y values indicate some location, and the weight represents a population of that location. The centroid that your program needs to compute is the weighted average of all of the x values, along with the weighted average of all of the y values. For example, let's say we have just two points.

Point #1: x = 4, y = 5, weight = 10
Point #2: x = 10, y = –1, weight = 20

We would compute the centroid as follows:

Weighted sum of x = 4*10 + 10*20 = 240
Weighted sum of y = 5*10 + (–1)*20 = 30
Sum of all the weights = 30 (and we would divide the x-sum and y-sum by this value…)
Centroid x = 240 / 30 = 8
Centroid y = 30 / 30 = 1
And we would conclude that the centroid is the point (8, 1).

Program hints: Tokenize each line of the input file. The delimiter is a space. The first token is x (latitude), the second token is y (longitude), and the third token is population. As you read input, you should maintain variables that keep track of the weighted sum of x and the weighted sum of y, and the total population.

To check your program, input.txt should give a centroid of (5.48, 4.21). According to your program, what is the centroid of the data from each of the other files?

usa.txt _____

canada.txt _____

australia.txt _____