**COSC 1336, Lab 6 Instructions, File Input, Output; Exceptions**

**Part 1:** In a previous lab, you created a function that accepted a number, and determined if it was a prime. Later, you created a function that computed the nth prime. These programs work fine if you have Python installed, but what if you don’t? Sometimes, no computer is available to run a program. In this case, it is better to have a table. A table can be viewed or printed easily.

Create a file containing a table showing the 1st 1000 prime numbers. Use your previous work to generate the primes, but output the data to a file. The output should look like this file:

<http://primes.utm.edu/lists/small/1000.txt>  
  
Notice that there are 10 primes per row, 100 rows. You will need to format the output. Remember that format() allows you to define how wide a field is. After every 10 primes, start a new row.

Your program should create a file called: **DDHH\_L6\_Lastname\_1000Primes.txt**You can save this part to the file: **DDHH\_L6\_Lastname\_part1.py**

**Do NOT submit the table on Blackboard! I will run your program and generate the table.**  
  
Save this part to the file: **DDHH\_L6\_Lastname\_part1.py**

**Part 2:** A typical computing job is to read data from a file, process the data, and output some results. In a previous lab, you calculated Austin temperature statistics. This was from data input at the keyboard. This portion asks you to do a similar task, but get data from a data file.

There are 50 states in the United States, each with its own population. Which state has the largest population? The lowest? What is the average population? What was the population of Texas? You are given a data file: **StateCensus2010.txt**. This has a list of all 50 states, in alphabetical order. It includes: the state name, abbreviation, and population. Scan this data, and output the following statistics:

* State with the maximum population
* State with the minimum population
* Average state population
* State of Texas population

Tip: Look at Program 6-14 on page 314 (4ed); page 262 (3ed); Program 7-14 on page 266 (2ed). It does input processing on a file similar to the file StateCensus2010.txt.

Save this part to the file: **DDHH\_L6\_Lastname\_part2.py**

**Part 3:** In part 2, you process a data file. You open the file and read each line of data in a loop. After reading each line, you look at the data and develop statistics. You also search for “Texas” and keep track of the population of Texas. This is great if everything works: the file opens OK; the data is correctly formatted, etc.

But what if something goes wrong? What if the file is not found, or if there is something wrong with the data? An exception is when something unexpected occurs. A good program will handle exceptions.

For this part, you will open a file, add up a list of numbers in the file, and print out the sum. If something goes wrong, it will be handled.

**Most of the work is already done**. Look at: Program 6-26 in the textbook on page 331 (4ed); page 279 (3ed); Program 7-26 in the textbook on page 283 (2ed). This code will: open a file, add up all the numbers, and print the result. It handles exceptions, but is unfriendly. It can only process one file and has poor error messages. It is not very nice.

Improve the program. Ask the user for the base file name, and then add the extension, “.txt” to the end. Add an outer loop, so the user can enter more file names (until <enter>) instead of only one. Change the error messages. Instead of unfriendly error messages like: “An error occurred”, tell the user why the error occurred, what the file name was, what line number the error occurred on, what data was in error. Test your program on the provided data files: data1.txt and data2.txt. Also test on a non-existent file, data3.txt.

Your program’s output should look like this:

**This program adds up a list of numbers in a file.**

**Enter file name without .txt extension. (enter nothing to quit): data1**

**Processing file: data1.txt**

**The values in the file data1.txt add up to: 6,600.66**

**Enter file name without .txt extension. (enter nothing to quit): data2**

**Processing file: data2.txt**

**Non-numeric data found in file: data2.txt at line: 3 with input: three hundred**

**Enter file name without .txt extension. (enter nothing to quit): data3**

**Processing file: data3.txt**

**A read error occurred on file: data3.txt**

**Enter file name without .txt extension. (enter nothing to quit):**

**>>>**

**Summary: Combine the following files into one: DDHH\_L6\_Lastname.py**

**and submit on the Blackboard assignment link.**

**DDHH\_L6\_Lastname\_part1.py (write 1st 1000 primes in a table)**

**DDHH\_L6\_Lastname\_part2.py (read census data, develop stats)**

**DDHH\_L6\_Lastname\_part3.py (read data, handle exceptions)**

**Extra credit: Enhance part 3. Add code to convert “three hundred” into 300, so textual data for numbers will work. Accept text for number names from “one” to “nine hundred ninety nine”. Assume a space between each word in the number. If NOT a valid number (digits or text) skip the invalid data and continue processing all valid numbers.**