**Project 2 Essay**

**Nicholas Wilkinson**

The video I found on Octave was long but informative. There were only a few commands that I had to look up from other sources, such as the “disp()” command. The main issue, which is only a problem due to the nature of the assignment, was that the publisher didn’t create a project that expanded as he taught Octave. Instead, he made several smaller programs. As I explained in my essay relating to the Octave tutorial assignment, I took his last idea, a function that created and plotted a BMI matrix, and made it much more complex. I made several nested functions and utilized user input to graph BMI, and on the graph had lines showing the boundaries for being very underweight, underweight, overweight and obese. The project only took me a while because I was still learning how to use Octave.

Transferring my plotter, salter and smoother programs into Octave was relatively smooth. I changed the algorithm of the plotter to be a sine function, with the only input being where to start and where to end. The program generates points every hundredth (going from 0 to 1 would have 101 values, including 0.01, 0.02, 0.03, and so on). The plotter and salter were done very quickly, however I ran into issues with the smoother. At first, I was looking for a built-in function to smooth the data for me. When I didn’t find one, I went to work converting my Java smoother into an Octave smoother. I only hit a roadblock when the smoother was adding null values into the matrix. After some digging and attempted fixes, I found the roadblock was of my own design. I had copied and pasted the section of code that smoothed the left side of the graph to utilize for smoothing the right side, but forgot to change one of the variables. This resulted in the right side scanning a smaller and smaller area to smooth before eventually having nothing to work with, and so returned a null object. Once this was fixed, however, the program worked perfectly. I also designed it so that the user can name the output file whatever they want. Additionally, the user does not need to input the file type, as the code concatenates “.csv” to the file name for them.

I explained what I did with the plotter, salter, and smoother reimplementation using third party libraries in a separate paper. To summarize, the plotter only took a while to find out what to use but was easy afterwards. The salter was the easiest, as all I had to do was change how the data was being given and stored. The smoother took the longest simply because there was nothing in the Apache Common Math API about moving/rolling average, smoothing data, or noise reduction in data. I ended up having to code a decent amount while using an Apache class as a helper.

The Monte Carlo Deck was easy enough, but still took me a while due to having to rewrite several methods to better suit the poker game. The method to evaluate what type of hand the player has took a while due to the complicated manner of checking the hand. Additionally, getting the user input took a while to fix, since it seemed to take the user’s input twice while asking about modifying their hand, leading to the player being unable to remove a second card from their hand. I designed several classes for the program: ‘Card’ to store the card’s suite and rank, ‘Deck’ to create, store and manage the deck, ‘Hand’ to create, store, manage and evaluate the hands, and ‘Poker’ to run the game. ‘MonteCarloCard’ was the class I used to run the Monte Carlo simulation and only contains a main method with a small amount of coding for loops and count storage. ‘PokerTest’ is the tester class I made to test the ‘Poker’ class, and is just a main method that has to manually be changed to either play the game with one or two players.

The paper on section 4.5, 4.6, and 4.7 from the book took a while, but was easy. The distributions aren’t hard at all, but the book has a weird way of explaining things sometimes, and so I had to reread some sections.

The database paper took a good amount of effort. I struggled to find a good database. When I found one and had made questions for it, I realized the data was just a collection of all of the different combinations for the attributes given. So, I instead used the data as a basis for creating my own randomly generated dataset, which is stored in a separate Excel file. ‘car.csv’ contains the original database, while ‘CarData.csv’ contains the values that I randomly generated. The program I made to generate the dataset is contained in ‘DatabaseCreation.java’. I also encountered some difficulty creating questions based on the data. For instance, I had one problem where my calculator could not compute an equation because the values being made were too big for it to handle. To solve this, I simply gave the formula as the answer.

The elements of this project tested me, and some were a struggle, but I enjoyed all of the assignments (outside of the chapter 4 essay). I’m glad I was able to get some more coding practice during my final semester, and I’m proud of my work.