ECE 368
Bryan Settles

I first started with this project by setting up the two sorts, insertion and bubble, as I had understood them, with the correct function parameters. This involved adding code in both my insertion and bubble sort to count the number of comparisons and moves the occurred in the sequence. After I started in this fashion, I wanted to be able to test these functions, although they were still yet to be improved. I created both the 'Save_File' and 'Load_File' functions using file pointers, fscanf, and fprintf, as I had done repeatedly in my past programming courses. I made sure to check the sample input and output to check my output formatting. These sorts worked, but took an extremely long time while sorting the larger text files.

After that, I began to attempt the changes to the sorts as had been described in the project instructions. I started with bubble sort by adding the gap sequence specified in the instructions. I also added a bool variable which described when a swap had occurred, and allowed my function to efficiently go about finding the next gap value and proceeding from there. After I tested this function and it seemed to be working, I continued and started to implement the changes to my insertion sort. At first, I found it difficult to create code to find the necessary sequence for implementing this code. I eventually got it working by separating the processes of finding the correct sequence, and sorting the long array based on that sequence. While it took additional time to find the sequence, it made up for it by increasing the sorting speed of my shell sort.

During the process of creating these two sorts, I added some helper functions to better separate it logically. These functions were used to swap values, find the position of the next gap, and to act as a comparator value for the built-in function 'qsort'. The swap and comparator functions were relatively straight forward, as I had written them before in ECE 264. In both of these functions, as well as many other places in my program, pointers were used, even in place of arrays at times, as a way to pass variables in between functions while being able to change the actual value of the variable. The gap function was used for the improved bubble sort, in order to divide the gap value by the value given in the instructions, 1.3. It also tested to see if the gap value was less than one, for the final gap value in the improved bubble sort sequence. All in all, this project has been a good way of practicing how to develop efficient code.