368 Project 1 Milestone 1 - Ethan Glaser

I have begun to make progress on project 1, but I have also started planning how I am going to implement the larger tasks included in the project. I have completed the file I/O functions (Load_File and Save_File) and have refamiliarized myself with some of the basic syntax and commands in C because it has been a while since I have done an assignment in C.

The larger tasks associated with this project are Shell Insertion Sort and Improved Bubble Sort, and these will take a bit more time and effort to implement. One of the challenges with creating a shell sort algorithm is generating the gap sequence. 2^p * 3^q is the required gap sequence, and the gaps must be implemented from highest to lowest, below the size of the data to be sorted. There are several ways to approach generating this sequence, but my current approach is evaluating each number N and below, where N is the size of the array. For each value, N is divided by 3 until the remainder is nonzero, and then that result is divided by 2 until the result is nonzero. If the final value is 1, the original number is part of the sequence. In order to increase the efficiency of the algorithm, I avoid creating an array of sequence values, instead iterating through and identifying the next value and performing shell sort using that value. Once I begin to test the sorting method, I will have a sense of whether this method is efficient. Other than that, the shell sorting will be relatively straightforward, using the examples that have been discussed in class for reference, but essentially creating new arrays for each gap value and sorting them using insertion sort. Comparisons and swaps will be monitored by incrementing the variables each time the respective event occurs.

The Improved Bubble Sort algorithm has less variability with the gap generation, as it just involved division and flooring. Other than that, the implementation will be similar to the shell sort, just switching insertion sort for bubble sort. I plan to implement a basic shell bubble sorting method that meets all of the requirements, evaluate its performance, and then identify where the efficiency can be increased to optimize performance. Most of the changes will probably be based on how the sub arrays are bubble sorted, as most of the rest of the sorting method is relatively set.