Report 1 – Ethan Glaser

This project consists of six functions – 2 for loading and savings arrays to and from files, 2 for savings sequences to files, and 2 sorting algorithms. The 4 loading and saving functions primarily consist of basic file i/o, either reading from a file (loadfile) or writing to a file (savefile, saveseq1, saveseq2). The sorting algorithms consist of a shell sort using a 3-smooth pattern for k with an insertion sort on each sub-array, and an improved bubble sort using a k pattern that decremented by a factor of 1.3.

The sequences for each sorting algorithm were generated in different ways. For the shell insertion sort, k was a 3-smooth pattern, which was generated by testing integers from the size of the input to 1, dividing by 3 until there was a remainder, and then dividing by 2 until there was a remainder. If the final value is 1 then the integer was used as a k value. This is a different, but more intuitive approach than that outlined in the project description, and it worked efficiently. The improved bubble sort sequence was generated in a more straightforward manner, beginning with a k value of the size / 1.3 and continuing to divide by 1.3 to generate the next k value.

As shown in the data tables, the time complexity, number of swaps, and number of comparisons are all related to the size of the data input. The time, comparisons, and swaps all increase by approximately the same rate as the size increases, although not exactly at the rate that the size increases.

The space complexity of the algorithms are extremely efficient, as the only memory being allocated is that of the initial array read into the loadfile function. No other memory is allocated throughout the project. The memory complexity is O(n) because the only time memory is allocated is for each integer in the original file.

Shell sort:

Size	Time	Comparisons	Swaps
1000	0.00	35266	4311
10000	0.00	615529	64818
100000	0.03	9484124	878713
1000000	0.41	135697411	11710260

Improved bubble sort:

Size	Time	Comparisons	Swaps
1000	0.00	36816	4399
10000	0.00	611916	62294
100000	0.02	9367379	813582
1000000	0.4	128013000	10071729