Mitch Bouma
ECE 368
HW # 6

Project 1 Approach

For the shell sort algorithm, the gap sequence specified is known at the Pratt Sequence. Pand q are the exponents to which 2 and 3 are raised to respectively. To implement this, I will find all products of powers of 2 and 3 no larger than the maximum (which is the number of the elements in the array). After getting an array of the gaps, I will then quicksort that array so that the gaps are decreasing. Finally, I will run insertion sort on all gaps in this sorted array.

My improved bubble sort will work similarly to the shell sort. In regular bubble sort, only items that are next to each other are compared (a gap of 1). However, rather than using insertion sort on the theoretical sub arrays for each gap, bubble sort will be used. Insertion sort moves an individual item multiple times in each pass whereas bubble sort will only swap once and then move on to the next item. The gap sequence will also be different. The number of items in the array divided by 1.3 will be the largest gap. Each successive gap will be divided by 1.3 again until a gap of 1 is reached.

The load function will simply read long ints from a file into an array and return a pointer to that array. The save function will work in a similar manner, printing from the array to a file. Fopen, fwrite, etc will be used.

In order to keep track of statistics such as swaps and comparisons, I will set up counters for each case. For example, when I check if an element is greater than another, the comparison counter will be incremented. The swap counter will only be incremented if that condition is verified and an element is moved. Also, since both functions will use temp variables, this will affect swap counts. Writing to a temp, rewriting to an array index, then writing from the temp to the other array index will count as 3 moves. I will account for this by adding increment statements in multiple places throughout the code. Run time must also be tracked, so finding the difference in time from when the algorithm is called to when it returns will give a sufficient value.