ECE 368 Project 1

Summary of functions:

For the load file function I used a temporary variable to store the number of elements in the original array inside the file on the first line. I used this number to allocate memory to array1 and then used a while loop to cycle through the file and put these long integers into array1.

For the save file function I used a counter and a for loop along with the fprintf function to save it to arr1 which writes to the user specified file.

For shell insertion the first thing I did was take care of the sequence. For this I made an array which contains all the possible sequences according to the user defined array. To get my sequences I used the Pratt Algorithm in which the next sequence number is generated by using previous numbers in the sequence array and multiplying them by 2's and 3's. Two variables (last_2ind, last_3ind) are used to keep indices that have already been multiplied by 2 and 3 and the array goes on from there. After my sequence array has been set up I use a while loop(outermost loop) to cycle backwards through the sequence array because it's in ascending order (we have start from the largest sequence value). Now inside this loop is normal insertion sort algorithm which is slightly modified so instead of comparing with neighbors it compares with values separated by the specific sequence we're running. The insertion sort also has two loops which is the outer loop and the inner loop.

For improved bubble sort I first used the logic described in the question to calculate the first gap to be used and then started a while loop so as to cycle through the gap values. This gap value is updated at the end of the loop using the last gap value in the same function used to calculate the first gap value. Inside this loop is regular bubble sort but it has been altered so that it compares value separated by the gap.

Analysis of Seq1 and Seq2:

Seq1: Time Complexity: O(nog(n)^2) Space Complexity: O(1)

Seq2: Time Complexity: O(n^2) Space Complexity: O(1)

Analysis of Run Times:

Type	**Size**	**#Moves**	**Time/sec**	**#Comparisons**
Insertion	1000000	197083699	1.090000	14594691
Bubble	1000000	20158664	1.000000	10079332
Insertion	100000	14004309	0.080000	1024455
Bubble	100000	1632360	0.090000	816180
Insertion	10000	960602	0.010000	71624
Bubble	10000	125212	0.010000	62606

Space Complexity:

Insertion Sort: O(1)

Bubble Sort: O(1)