Selection Sort		
List Size	**Comparisons**	**Time (seconds)**
1,000 (observed)	499500	0.1381380558013916 s
2,000 (observed)	1999000	0.5572471618652344 s
4,000 (observed)	7998000	2.2449445724487305 s
8,000 (observed)	31996000	9.216675996780396 s
16,000 (observed)	127992000	36.5636465549469 s
32,000 (observed)	511984000	143.05527687072754 s
100,000 (estimated)	4999950000	1428.265625 s
500,000 (estimated)	124999750000	36750 s
1,000,000 (estimated)	499999500000	143000 s
10,000,000 (estimated)	49999995000000	14300000 s

Insertion Sort		
List Size	**Comparisons**	**Time (seconds)**
1,000 (observed)	247991	0.2417926788330078 s
2,000 (observed)	1018723	0.8247733116149902 s
4,000 (observed)	3995271	3.239780902862549 s
8,000 (observed)	16112202	13.406851768493652 s
16,000 (observed)	64667457	52.794450521469116 s
32,000 (observed)	257507127	210.0624861717224 s
100,000 (estimated)	2526072539	2062.283203 s
500,000 (estimated)	63151800000	51500 s
1,000,000 (estimated)	252600000000	206000 s
10,000,000 (estimated)	25260000000000	20600000 s

1. Which sort do you think is better? Why?

   Insertion sorting is better, because it makes less comparisons.

2. Which sort is better when sorting a list that is already sorted (or mostly sorted)? Why?

   Insertion, because it will have an efficiency measurement of O(n) compared to selection sort that will be $O(n^2)$. Selection sort is $O(n^2)$ because it must compare each value to every other value regardless of its positions pre-sorted.

3. You probably found that insertion sort had about half as many comparisons as selection sort. Why? Why are the times for insertion sort not half what they are for selection sort? (For part of the answer, think about what insertion sort has to do more of compared to selection sort.)

Lab 6 – Ashley Sutter

Insertion sort took longer to execute, because the swapping operation is within a nested for-loop which has a Big O of n^2 while selection sort's swapping operation is only in the outmost for-loop with a Big O of n.