Selection Sort		
List Size	**Comparisons**	**Time (seconds)**
1,000 (observed)	499500	0.1264200210571289
2,000 (observed)	1999000	0.3767542839050293
4,000 (observed)	7998000	1.7542829513549805
8,000 (observed)	31996000	6.234943151473999
16,000 (observed)	127992000	36.67119121551514
32,000 (observed)	511984000	138.8613269329071
100,000 (estimated)	4999843750	804.5433489
500,000 (estimated)	1.249 * 10^11	19477.4929466
1,000,000 (estimated)	4.999 * 10^11	82952.3556821
10,000,000 (estimated)	4.999 * 10^13	829523.5568212

Insertion Sort		
List Size	**Comparisons**	**Time (seconds)**
1,000 (observed)	247986	0.12298393249511719
2,000 (observed)	1018717	0.5829510688781738
4,000 (observed)	3995264	1.8346247673034668
8,000 (observed)	16112194	7.034783840179443
16,000 (observed)	64667449	29.893370866775513
32,000 (observed)	257507119	120.1005392074585
100,000 (estimated)	2514717959	1.3252 * 10^17
500,000 (estimated)	6.287 * 10^10	3.0162 * 10^18
1,000,000 (estimated)	2.518 & 10^11	1.2698 * 10^19
10,000,000 (estimated)	2.518 * 10^13	1.2698 * 10^21

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

   I think the insertion sort is better because the insertion sort is stable and efficient. The cons to the selection sort algorithm is that it always performs a quadratic number of comparisons, and it moves each element once at most, which is not super efficient. Although both algorithms' time complexity is O(n^2) in the average case, the insertion sort complexity is O(n) in the best case, whereas selection is O(n^2).

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

   Insertion sort is better when sorting a list that is already sorted (or mostly sorted) because that will lead to the best case time complexity, O(n), when the best case time complexity for the selection sort is still O(n^2).

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.)

   Insertion sort works by comparing two elements at a time, which is why it has about half as many comparisons as selection sort. Compared to insertion sort, selection sort only picks the minimum element from the whole list and sorts it each time, which leads to more comparisons. As they are both O(n^2), they will take a similar amount of time. Selection sort has to find the index of the minimum, swap, and iterate through the array, while insertion sort has to spend time inserting values to sorted subarrays and comparing those values.

	Number of Quicksort Comparisons	
Starting List	pivot = first	pivot = median of 3
Ordered, ascending		
n = 100	4950	480
n = 200	19900	1153
n = 400	79800	2698
n = 800	319600	6187
Random		
n = 100 (average 10 runs)	662	552
n = 200 (average 10 runs)	1599	1320
n = 400 (average 10 runs)	3582	3014
n = 800 (average 10 runs)	8643	7135
Observed Big O() behavior, ordered with pivot = first : O(n^2)		
Observed Big O() behavior, ordered with pivot = median of 3 : O(n^2)		
Observed Big O() behavior, random with pivot = first : O(n log n)		
Observed Big O() behavior, random with pivot = median of 3 : O(n log n)		
For random list, observation regarding using first vs. median of 3 : Both are O(n log n) but the median of 3 has less comparisons.		