Selection Sort		
List Size	**Comparisons**	**Time (seconds)**
1,000 (observed)	499,500	0.0974
2,000 (observed)	1,999,000	0.4202
4,000 (observed)	7,998,000	1.59
8,000 (observed)	31,996,000	6.28
16,000 (observed)	127,992,000	26.43
32,000 (observed)	511,984,000	116.64
100,000 (estimated)	4,999,950,000	1,026.13
500,000 (estimated)	124,999,750,000	25,653.41
1,000,000 (estimated)	499,999,500,000	102,613.76
10,000,000 (estimated)	49,999,995,000,000	10,261,384.79

Insertion Sort		
List Size	**Comparisons**	**Time (seconds)**
1,000 (observed)	247,986	0.1898
2,000 (observed)	1,018,717	0.8072
4,000 (observed)	3,995,264	3.27
8,000 (observed)	16,112,194	12.71
16,000 (observed)	64,667,449	55.05
32,000 (observed)	257,507,119	210.77
100,000 (estimated)	2,499,975,000	2,012.19
500,000 (estimated)	62,499,875,000	50,305.03
1,000,000 (estimated)	249,999,750,000	201,220.33
10,000,000 (estimated)	24,999,997,500,000	20,122,050.72

1. Which sort do you think is better? Why?
   All else equal selection sort. Selection sort had roughly double the number of comparisons but took half the amount of time. Since tracking the number of comparisons takes memory, there may be a time versus memory tradeoff. But my analysis was done with a list of random numbers, i.e., the list was not sorted. So, in the case where my list is partially sorted, insertion sort may be more time efficient.

2. Which sort is better when sorting a list that is already sorted (or mostly sorted)? Why?
   Insertion sort! The closer the list is to being fully sorted, the closer insertion sort becomes to big O of n. The number of comparisons for insertion sort will become even smaller while the number of comparisons for selection sort will stay the same. When insertion sort makes less comparisons it also will make less swaps reducing the time complexity.

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 does not necessarily need to run through the entire list for each iteration. On average it will run through half the list. On the other hand, selection sort looks at each element in the list for each iteration. As for the time component, insertion sort swaps each time a comparison is met while selection sort makes this swap just once for each iteration.