Lab 6

Selection Sort		
List Size	**Comparisons**	**Time (seconds)**
1,000 (observed)	499,500	0.0506129264831543
2,000 (observed)	1,999,000	0.1850588321685791
4,000 (observed)	7,998,000	0.745549201965332
8,000 (observed)	31,996,000	3.01857304573059
16,000 (observed)	127,992,000	11.974913120269775
32,000 (observed)	511,984,000	48.802639961242676
100,000 (estimated)	4,999,950,000	506.129264831543
500,000 (estimated)	124,999,750,000	12,653.2316208
1,000,000 (estimated)	499,999,500,000	50,612.9264831543
10,000,000 (estimated)	49,999,995,000,000	5,061,292.64831543

Insertion Sort		
List Size	**Comparisons**	**Time (seconds)**
1,000 (observed)	246,992	0.044595003128051 76
2,000 (observed)	1,016,724	0.17235612869262695
4,000 (observed)	3,991,272	0.645949125289917
8,000 (observed)	16,104,203	2.793045997619629
16,000 (observed)	64,651,458	11.099189043045044
32,000 (observed)	257,475,128	41.47763681411743
100,000 (estimated)	2,469,920,000	445.9500312805176
500,000 (estimated)	61,748,000,000	11,148.75078

1,000,000 (estimated)	246,992,000,000	44,595.00312805176
10,000,000 (estimated)	24,699,200,000,000	4,459,500.312805176

1. Which sort do you think is better? Why? I think that the insertion sort is better than the selection sort. In class, we analyzed the worst case, best case, and average case amount of comparisons for each type of sort. The amount of comparisons for a selection sort will always have an O(n^2) time complexity. On the other hand, the amount of comparisons for an insertion sort will have a O(n) time complexity in the best case scenario.

2. Which sort is better when sorting a list that is already sorted (or mostly sorted)? Why? The insertion sort is better when sorting a list that is already sorted or mostly sorted because only one comparison will need to be made for each pass.

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 compared to selection sort.) The insertion sort only compares as many elements as it needs to in order to place the next element. On the other hand, the selection sort must scan all remaining elements to find the next element. This is why the insertion sort has about half as many comparisons as the selection sort. The times for the insertion sort are not half of what they are for selection sort because the insertion sort has to do more shifts compared to the selection sort. The selection sort does swaps.