Project 1 sorting:

# OF ITEMS	SELECTION SORT TIME	QUICK SORT TIME
10	0.001035086 seconds	4.81177E-4 seconds
100	0.011783277 seconds	0.002057287 seconds
1000	0.655331640825 seconds	0.0315778765 seconds
5000	66.1141574781 seconds	0.553782914 seconds

Project 2 sorting:

# OF ITEMS	SELECTION SORT TIME	QUICK SORT TIME
10	6.73148E-4 seconds	5.578E-5 seconds
100	0.00167047 seconds	7.4136E-4 seconds
1000	0.092903854 seconds	0.01177297 seconds
10000	6.585634986 seconds	0.188281616 seconds

For both, the quickness of quickSort was clearly visible, as each time quickSort did the sorting faster than selectionSort. However, the relationships described in class didn't really seem to apply to these programs–the ratios between each number are inconsistent. For example, for selection sort, the time didn't seem to increase as N^2. It may have been due to the machine I was running it on.

Project 1 searching:

# OF ITEMS	SEQUENTIAL SEARCH TIME	BINARY SEARCH TIME
10	1.7263E-4 seconds	5.7066E-5 seconds
100	2.55134E-4 seconds	4.2169E-5 seconds
500	6.19989E-4 seconds	4.9494E-5 seconds

Project 2 searching:

# OF ITEMS	SEQUENTIAL SEARCH TIME	BINARY SEARCH TIME
10	2.9635E-5 seconds	4.0846E-5 seconds
100	6.6337E-5 seconds	3.8891E-5 seconds
500	2.51555E-4 seconds	3.901E-5 seconds

For both projects, binary search tended to work faster. There were some instances were sequential search was faster (like for 10 items in project 2), which could happen if the algorithm was lucky enough to find it quicker than binary search. However, like the sorting algorithms, the search algorithms didn't really function as O(N) and O(log2N) respectively, again probably due to the machine I was running the program on.