Cora Coleman
Writeup, Algorithms lab

1. In order to analyze the rate of growth of SelectionSort using the doubling rule, I found the average difference in runtime for each dataset between the ranges of N = 2000, 4000, 8000, 16000. I therefore determined the average time SelectionSort took to double over that range to be 0.423 for the randomized movie score dataset, 0.426 for the reversed movie score dataset, and 0.185 for the sorted movie score dataset. This supports the faster times the sorted dataset returned on average, over all the datasets. However, on average the random dataset returned slower times than the reversed dataset. This differs from how the reversed dataset is slower on average to double than the random dataset. Perhaps this is because of the nature of the reversed dataset, which requires a constant number of steps for sorting, whereas the random dataset could be any number of orderings, of which on average require more steps to sort.
2. According to the algorithm, SelectionSort uses (N– 1) + (N– 2) + ... + 1 + 0 ~ N^2/2 compares and N exchanges.
3. The order of the input data could be immediately determined as sorted using any of the code's helpful methods for debugging and determining whether an array is sorted and even if it is sorted within a particular index range (isSorted() methods). The methods less() and exch() are called each time a new, unsorted value is encountered and determined to be less than the current value. The method less() is used more than exch() because while every unsorted value must be evaluated to see if it is less than the current value, it will not always be and therefore will not need to be exch(). The method exch() would be used minimally in a pre-sorted dataset whereas in a random or reversed dataset it would be used more often to complete a sort.