**Algorithms Report**

1. Which algorithms run mostly in quadratic time, i.e. O(n^2)?

Algorithms 1, 2, and 3 run (relatively) in quadratic time for By Name and By ZipCode comparators. Algorithms 2, and 3 also run in quadratic time for the By Longitude comparator.

1. Which algorithms run mostly in O(n.log\_n) time?

Algorithms 0 and 5 run in n.log\_n time for By ZipCode and ByName comparators. Algorithms 0, 1, and 5 run in n.log\_n time for the By Longitude comparator.

1. Which algorithms use the functional style, using Cons lists?

Algorithm 1 and 4 runs using the Cons list style.

1. Which algorithm is the *selection* sort?

The unmentioned heapsort is a part of the selection sort family.

1. Why is there a difference when the algorithms use a different Comparator?

The algorithms runtime depends on the complexity of the objects that they are comparing, thus more complex objects result in more runtime (notice how string objects take longer to sort using the algorithms than integers).

Summary of Results:

From running many different sort algorithms, it is apparent that not all sorting methods are made equal; that is all sorting methods accomplish their purpose, however some are more efficient than others. Take, for example, on small inputs nearly all the sorting methods have similar runtime. However as the size of the inputs increase, the O runtime of the algorithms result in drastic changes in the efficiency of the program. Namely, quadratic runtime is less efficient than logarithmic runtime, thus resulting in algorithms such as algorithm 0 being significantly quicker in producing output than algorithms such as algorithm 3. Selection Sort algorithms are faster than quadratic runtime algorithms, but quicksort and linear algorithms are even quicker. So when designing code, it is not enough to just make the program run; the code must be efficient, otherwise it may result in horrible runtime or, even worse, stack overflow error.