Samuel Boehle Final Project Report

Part A:

The tests I have run conclude that a hash table is significantly faster in insert and search times than a doubly linked list. This is because in order to insert in a linked list the program must traverse the entire linked list until it finds the end. This results in an insertion time that is scaled linearly with the size of the linked list. The search of a linked list is also scaled linearly but faster than insert. This is because the average search is going to be in the middle of the list, so the program only needs to traverse half of the linked list on average. In comparison a hash table is significant faster. As long as the size of the table is large enough to not cause that many collisions, then the performance should be close to linear since all the program has to do is map a value to an index. The obvious choice is to choose a hash table in order to save on system resources for the medical company.

Part B:

The bubble sort scales exponentially since as more numbers are sorted, the algorithm needs to make more operations in terms of swapping two of the numbers, and on the total amount of times the algorithm needs to iterate through the list. The heap however scales linearly with its size. This is because the heapify algorithm grows at a consistent rate as it scales, and the insert function into the heap is very efficient by only needing to be completed a max of time for the number of levels the heap currently has. The best choice for the medical company would be to use heap sort as it is much more efficient than bubble sort especially if they are going to be sorting large amounts of data. Another plus side to heap sort is that the amount of memory required to store the data should be the same as a bubble sort since in both cases the data being sorted is being stored in an array.