Daniel Weitman
COSC-320

(a)
n = Array Size

Quick Sort
Best Case: Omega(n*log(n))
Worst Case: O(n^2)

Merge Sort
Best Case: Omega(n*log(n))
Worst Case: O(n*nlog(n))

(b)
The absolute timing complexity scales linearly with number of element and/or element sizes. The absolute timing complexity lines up with the theoretical timing complexity.

(c)
Similar to the O(n^2) sorts, the merge and quick sorts had no intersections between them for array sizes tested. Therefore, the merge sort performed faster for all tests. Yet, unlike the O(n^2) sorts the merge and quick sorts had a linear trend showed in their graph. This means that the time discrepancy was growing as array size increased but not to as high a degree. So both algorithms were comparable to each other in sort time for all times tested even though merge was slightly faster in all instances. Quick sort had a few instances where its n^2 worst case was shown, merge sort was always seemed to run as expected with nlog(n).

(d)
Yes, quick sort had a wide discrepancy in runtime for similar sized and populated arrays. Mergesort divided the arrays and then re-combined them ensure nlog(n) is always its time even in the worst case.

(e)
The implementation of Quick and Merge sort are much more involved than bubble as it both have a more sophisticated algorithm to optimize runtime.

(f)
Other than adding more sorting algorithms for different cases the code is executed well. Additionally, adding some more input validation may be useful in some instances.