WRITEUP

1) Given the worst-case scenario, the time complexity of bubble sort is n^2
2) Given the worst-case scenario, the time complexity of bubble sort is n*log(n)
3) In the very worst case, the time complexity of quicksort is n^2. However, the pivot value can be adjusted such that the worst-case time complexity is n*log(n)
4) Time complexity depends on the values in the gap sequence. Worst case based on that is n*(log(n)) * (log(n)).
5) For each of the sorts above, the constant doesn't impact the time complexity significantly since it's the lowest order term in the best and worst-case time complexities for each of the sorts. The constant for all of them is affected by the size of the array to be sorted.

What I learned:

- Learned how shell sort, quick sort, and binary-insertion sort worked.
- Learned about the different time complexities of those sorts and how they are affected by minor tweaks in the algorithm implementation.
- Learned why certain sorts like quicksort and binary-insertion sort are better by creating and performing their implementation in code.

Experimentation:

- Experimented with quicksort by changing the pivot value from the leftmost value in the array(based on what was in pseudocode) to the median of leftmost, rightmost and center value to see how it affects the number of swaps needed to be made. Learned that worst-case time complexity is improved by changing it to that by observing the difference in swaps made for the given array
- Experimented with shell sort by playing around with the formula for generating the gap sequence. The original sequence(based on assignment PDF) was 5n/11. Changed it to 3n/4 to see the effect on time complexity. With the new gap sequence, significantly fewer comparisons were made and slightly fewer swaps were made.