Alexander Shah
Homework 8: Sorting
EN.605.202.81 Section 84

1a.
Simple insertion on a sorted file requires comparisons between all elements, but none are moved, so n-1 comparisons and 0 changes.

1b.
Simple insertion on a reversed file requires (n*n-1)/2 comparisons to look at all elements, swap them, then compare again, and (n*n-1)/2 swaps to change their positions repeatedly until sorted

1c.
Simple insertion on a semi ordered file could require as many comparisons and insertions as a random file, up to n^2.

2a.
Shell sort on a sorted file with increments 2 and 1 will require n/2 + n comparisons and 0 changes because nothing needs to move

2b.
Shell sort on a reversed file with increments 2 and 1 will require just as many comparisons initially as sorted, n/2 + n, but will also require moving values, so there will be an additional n comparisons and changes accordingly. So there will be n/2 + n^2/2 comparisons and interchanges.

2c.
Shell sort on odds and evens file will perform n/2 comparisons when the gap is 2, and then when gap is 1 it will perform like a worst case insertion sort, so n^2 comparisons and changes.

3a.
When we merge ordered files where every element of a and b are alternating, we must compare every element as we march through the lists, so the number of comparisons is m+n-1.

3b.
When merging two sequential lists the algorithm uses n comparisons, because while every comparison will be that an element in A is less than the first in B, the algorithm must still iterate through the lower list to merge them.

4a.
Merging lists such that B is between the left half and right half of A requires comparing every element up til the half way point (n/2) in A compared to the first value of B, and then once B is less than the right half of A, the comparisons continue to compare B values against A[n/2+1] until B is exhausted, then the remaining portion of A can be merged in. So it will take n/2+m comparisons, the length of the left half of A plus B.

4b.
When B only has one value and it’s less than anything in A, only one comparison needs to be made to determine that B[1] goes first, then the rest of the A array is added.

4c.

In the case B has a single element greater than the values in A, the merge must compare B[1] against every value of A until it determines that it goes last, so n comparisons.