Module 5 Problems

1. [15 pts] Suppose we break an input into 3 evenly sized subcollections. Put the sorted components back using a 3-way merge.
What is the complexity of this algorithm?

2. [15 pts] Suppose we have an instance of quicksort that splits evenly every other time, with the alternate instances being partitioned into 1,k-1 [the worst case]. That is, the worst case occurs every other recursive call. What is the complexity of quicksort under these circumstances?

3. [25 pts] Suppose you are an accountant (sorry), and you have records with 2 fields to sort on:

1) Transaction # and
2) Transaction Date.

Transaction #'s and Dates will tend to correlate, but the correlation is not foolproof. Given a list that has already been sorted by transaction #, which sorting algorithm would you choose to sort the list by date: quicksort or insertion sort? (As always, justify your answer)

4. [20 pts] Suppose you have k sorted arrays of size n that you wish to combine into 1 sorted array of size kn. You use merge in order, merging array_1 with array2, merge that array with array 3, ..., and so on in sequence. What is the complexity of this n-way merge? (Big Oh notation encouraged).

5. [25 pts] Like Problem 4, you have k sorted arrays of size n to merge. This time, however, you pair 2 of the arrays and merge them, repeating this process until there are k/2 sorted arrays of size 2n. Now, combine pairs of these 2n-sized arrays and so on until 1 array remains. What is the complexity of this merge operation?