Problem 1. For this problem, assume that you have access to a procedure $\text{Merge}(A, B, C, m, n)$, that merges two sorted arrays $A[1..m]$ and $B[1..n]$ into a single sorted array $C[1..m+n]$. This procedure runs in $m + n - 1$ comparisons. Assume we also have access to unlimited dynamic array allocation.

Suppose that you have an $n \times n$, 2-dimensional array $D[1..n,1..n]$, such that each row is sorted (i.e. $D[i,j] \leq D[i,j+1]$). We want to compute a single sorted array that contains all $n^2$ elements.

The first strategy for accomplishing this is called cascading merging. First merge the first two rows $D[1,*]$ with $D[2,*]$ to form an array of length $2n$, then merge the third row $D[3,*]$ with this array to form an array of length $3n$, etc. The final merge is between the last row $D[n,*]$ with an array of length $(n-1)n$ to form the final sorted array of length $n^2$.

The second strategy is called balanced merging. Assume that $n$ is a power of 2. First merge pairs of rows $D[1,*]$ and $D[2,*]$ together, $D[3,*]$ and $D[4,*]$ together, and so on until merging $D[n-1,*]$ and $D[n,*]$ together. The result is $n/2$ sorted arrays each of size $2n$. Next, repeat the balanced merging on these arrays, resulting in $n/4$ arrays each of size $4n$. This is repeated until there is one array of size $n^2$.

(a) (i) What is the exact number of comparisons for cascading merging? Justify your answer.
    (ii) What is the high order term?

(b) (i) What is the exact number of comparisons for balanced merging? Justify your answer.
    (ii) What is the high order term?

(c) Which algorithm has a better high order term, cascading merging or balanced merging (or are they the same)?

Problem 2. Assume merging two lists of sizes $m$ and $n$, where $m \leq n$, takes exactly $\lceil m \log_2(4n/m) \rceil$ comparisons.

(a) What is the number of comparisons for cascading merge? Just get the high order term exact. Justify your answer. NOTE: $\log(n!) \sim n \log n$.

(b) What is the number of comparisons for balanced merging? Just get the high order term exact. Justify your answer.

(c) Which algorithm has a better high order term, cascading merging or balanced merging (or are they the same)?

Problem 3.

(a) Recall the insertion-sort-like algorithm in Problem 4 from Homework 2, where you know that certain pairs of contiguous items in the sorted order are contiguous in the input. Write the pseudo-code for this algorithm, where the outer loop is done by recursion. Assume that $n$ is even.

(b) Write a recurrence for the number of comparisons in the worst case (assuming that $n$ is even).