Problem 1. Imagine there is an algorithm, X, whose runtime follows the following recurrence equation:

$$T(n) = 9\,T\left(\frac{n}{3}\right) + n, \qquad T(1) = 1$$

Answer the following questions:

1. Draw a recursion tree showing the top two and the bottom two levels of the recursion tree.
2. Solve the recurrence equation using the recursion tree exactly.
3. How many subproblems are in level j, somewhere between the top and the bottom level of the recursion tree?
4. What is the size of each subproblem at level, j?
5. What is the value of $T(9)$?

Show your work.

Problem 2. For this question we will work with a slightly modified merge sort algorithm. In the class we saw how merge sort would divide the input array all the way to a single element array before we merge them. However, in modified version of merge sort we would split the arrays only up to the point where we have k subarrays, each of size, n/k. We would use insertion sort to sort each one of those k subarrays and then merge them using the merge subroutine. Consider, this level with k subarrays as the base case for this modified merge sort algorithm. Just like in the merge sort, if we merge two subarrays whose combined length is n, it takes $n-1$ comparisons. Answer the following questions:

1. Draw a recursion tree, showing the top four and the bottom two levels.
2. What is the worst case number of comparisons to sort k subarrays each of size n/k using insertion sort (with sentinel)?
3. Solve the recursion tree for the worst case number of comparisons using merge sort. What is the base case?
4. What is the exact number of comparisons for the modified merge sort algorithm?
5. Is it better than the regular merge sort algorithm?
6. What happens when $k = n$?
7. What happens when $k = 1$?