Name & Section Number: _____

1. (a) Describe briefly but clearly how to efficiently merge three sorted lists, of sizes $m_1 \leq m_2 \leq m_3$, into a single sorted list. Try to minimize the number of comparisons under the assumption that $m_1 \approx m_2 \approx m_3$, but no proof of optimality required.

   (b) Exactly how many comparisons does your algorithm use in the worst case (as a function of m_1, m_2, m_3).

2. Consider a MergeSort-like algorithm in which the list of size N is split into three equal sized lists, each list is sorted recursively, and the three lists are then merged into a single sorted list.

   (a) Write pseudo-code for your algorithm. You may assume a three-way merge algorithm is available (as described in Question 1). Your algorithm should work even if N is not nice.

   (b) Analyze exactly how many comparisons your algorithm uses by drawing out the tree and summing across the rows (as we did in class for standard mergesort). You may assume N is nice (i.e., N is a power of 3).

(c) Write a recurrence for exactly how many comparisons your algorithm uses under the assumption that N is nice (i.e., N is a power of 3).

(d) Write a recurrence for exactly how many comparisons your algorithm uses without assuming N is nice.