Name & Section Number: ______________________________

1. We are going to derive a good upper bound on the worst case number of comparisons in (standard) heapsort. In order to simplify the algebra assume that the size of the list that you are sorting is one less than a power of two (i.e., a full tree). Thus, an array of size $n = 2^k - 1$ is associated with a binary tree with k levels numbered $0, 1, \ldots, k-1$.

   (a) How many nodes are on level j?

   (b) How many comparisons does the heap formation phase use to sift an element rooted on level j in the worst case.

   (c) Give a summation for the total number of comparisons for heap formation.

   (d) Simplify the summation. Show your work.

(e) How many comparisons does the remainder of heapsort use for each sift after removing an element from level j? Note that the tree gets smaller so not all elements on level j have exactly the same number of comparisons.

(f) Give a summation for the total number of comparisons for the second phase of heapsort.

(g) Simplify the summation. Show your work.

(h) Add the two totals and simplify.

2. Consider an array of size eight with the numbers $80, 30, 40, 70, 10, 20, 60, 50$. Assume you execute quicksort using the version of partition from CLRS. Note that in this algorithm an element might exchange with itself (which counts as one exchange).

   (a) Show the array after the first partition. How many comparisons and exchanges are used?

   (b) Show the left side after the next partition. How many comparisons are used? How many exchanges?

   (c) Show the right side after the next partition on that side. How many comparisons are used? How many exchanges?

3. In class (on Friday) we will analyze quicksort assuming that it always splits at the $n/4$ position (or $3n/4$ position). Analyze quicksort assuming that it always splits at the $n/3$ position (or $2n/3$ position). (Use Constructive Induction to get an upper bound.)