Due at the start of class Wednesday, June 20, 2012.

**Problem 1.** Consider an array of size eight with the numbers 30, 80, 50, 60, 20, 10, 70, 40. Assume you execute quicksort using the version of partition from CLRS. [Note that an element can exchange with itself, which counts as one exchange.]

(a) What is the array after the first partition. How many comparisons did you use? How many exchanges?

(b) Show the left side after the next partition. How many comparisons did you use? How many exchanges?

(c) Show the right side after the next partition on that side. How many comparisons did you use? How many exchanges?

(d) What is the total number of comparisons in the entire algorithm? What is the total number of exchanges in the entire algorithm?

**Problem 2.** Assume you execute quicksort using the version of partition from CLRS. We are interested in the exact fewest number of comparisons quicksort will do as a function of $n$ (i.e., its best case)? Assume $n = 2^k - 1$ (where $k$ is a natural number).

(a) (i) Give an example with $n = 3$ that does as few comparisons as possible.

(ii) Give an example with $n = 7$ that does as few comparisons as possible.

(iii) Give an example with $n = 15$ that does as few comparisons as possible.

(b) We might guess that the number of comparisons is approximately $n \lg(n+1) - 2n$. (Why?) Create a table with a column for $n = 1, 3, 7, 15$; a column with the exact number of comparisons for quicksort in the best case; a column with the value of the approximate guess, $n \lg(n + 1) - 2n$; and a column with the difference between the exact value and the approximate guess.

(c) Using the information from the above table give an exact formula for the number of comparisons.

**Problem 3.** Assume you have a list of $n$ elements where the first $n/k$ elements are the smallest (but not sorted), the next group of $n/k$ elements are the next smallest (but not sorted), ..., and the last $n/k$ elements are the largest (but not sorted). You may assume $k$ divides $n$.

(a) Give an algorithm that sorts this list with as few comparisons as possible (as a function of $n$ and $k$). Just get the high order term right. How many comparisons does your algorithm use?

(b) Show that your algorithm is optimal using a decision tree argument on the *entire* list. (I.e., do not argue that you must solve $k$ independent sorting problems.)