Due at the start of class Thursday, October 21, 2004.

**Problem 1.** Consider an array of size eight with the numbers in the following order 40, 20, 80, 60, 30, 10, 70, 50.

(a) What is the array after heap formation (BUILD-HEAP)? How many comparisons does the standard algorithm use?

(b) Starting with the heap, show the array after each element sifts down during the remainder of heapsort. How many comparisons does the standard algorithm use for each sift? What is the total number of comparisons for all of the sifts?

(c) How many comparisons does the modified algorithm (Floyd’s version) use for heap formation (BUILD-HEAP)?

(d) Starting with the heap, what is the total number of comparisons that the modified algorithm (Floyd’s version) uses for the remainder of heapsort.

**Problem 2.** A d-ary heap is like a binary heap, but instead of two children, nodes have d children.

(a) How would you represent a d-ary heap in an array?

(b) What is the height of a d-ary heap of n elements in terms of n and d.

(c) Explain loosely (but clearly) how to extract the maximum element from the d-ary heap (and restore the heap). How many comparisons does it require?

(d) How many comparisons does it take to sort? Just get the high order term exactly, but show your calculations.

(e) What value(s) of d are optimal? Justify your answer.

**Problem 3.** Assume you are given a reverse heap of size n (where the smallest element is on top) stored in an array in the standard way, and you are also given a real number x. Design an algorithm to determine whether the kth smallest element in the heap is less than or equal to x. The worst-case running time of your algorithm should be O(k), independent of the size of the heap. Notice that you do not have to find the kth smallest element; you need only determine its relationship to x.

You must explain your algorithm (briefly and clearly) in English. You do not need to give code.