Assignment 3. **Heapsort and quicksort.**

Given February 4, due Monday, February 16.

1. The quick select selects the largest x fraction of the elements of an array in $O(n)$ time for an array of size n. Here x is a number between 0 and 1. Now, suppose there are k such numbers $0 < x_1 < \cdots < x_k < 1$ and we want to move the elements of the array a so that the x_1 fraction comes first, then the $x_2 - x_1$, and so on. For example, suppose $k = 9$, $x_1 = .1$, $x_2 = .2$, and so on. We want the array reordered so that the top 10% of the numbers come first, then the second 10%, and so on. Within these 10% groups, the numbers need not be sorted. Find an algorithm based on the quick select idea that is better than sorting if $1 << k << n$.

2. (from CLR, page 151) Give an $O(n\log(k))$ algorithm to merge k sorted lists into a single sorted list of length n. *Hint*: Make a heap of the largest element of each of the k sublists not yet merged into the final list.

3. Given an array of length n, not in order, we want to determine whether any two elements are the same. The algorithm must use only binary comparisons. Is it possible to do this using fewer than $O(n\log(n))$ comparisons or equality tests? We know that it is not possible to sort this list in fewer than $O(n\log(n))$ comparisons.