Homework assignment 1, due Thurs Sep. 13 (one week later for FEEDS students)

Look up course webpage under "Homeworks," specifically for

- late submission policy which you must follow strictly – otherwise you could lose all points;
- cheating policy and bonus policy under "about homeworks."

Most answers should contain at least the following 4 parts. (Certain problems may require more, and a very few, less).
- The idea of the algorithm – you could choose to illustrate this with an example. Mention familiar design techniques used, if any.
- A pseudocode of the algorithm. Any segment of code that takes constant time may be abbreviated by one instruction. Any algorithm done in class or in the assigned readings can be quoted. All else must be explained.
- A proof of correctness: argue why the desired output is produced when the algorithm halts, and that it does halt.
- An analysis of the worst case time complexity, in terms of the relevant input size parameters.

1. **(Compulsory, 15 points)** The *multiplicity* of an element v in an (unordered) sequence S of numbers, is the number of times v appears in S. The *mode* of S is that element with the maximum multiplicity in S.
   Design an $O(n \log(n/m))$-time algorithm for the following problem specification.
   Input: An unordered sequence S of n elements, whose mode has multiplicity m (the value of m is *not* given as part of input).
   Output: the mode element in S and the value of its multiplicity, i.e, the value of m.

2. **(Compulsory 15 points)** Design the most efficient algorithm you can for the following problem.
   Input: A heap with the smallest element on top, and a value x.
   Output: 'yes' if x is less than or equal to the k^{th} smallest element in the heap.

   Note: k and n are independent parameters.

3. **(Compulsory, 25 points)**

   - Let $\{x_1, \ldots, x_n\}$ be a set of n arbitrary distinct numbers where $n = pq$. We wish to partition the set into q subsets $S_1, \ldots, S_q$ each of size p, such that if $i < j$, then every integer in S_i is smaller than every integer in S_j. Show this can be done in $O(n \log q)$ comparisons.
   - Consider the above problem when all numbers were all positive integers smaller than some polynomial in n? What is the best you can do? And why?
   - What is the most efficient algorithm you can think of for sorting n positive integers smaller than some polynomial in n?

4. **(Compulsory, 25 points)** PROBLEM (NOT EXERCISE) 14-1 in Book.

5. **(Optional 20 Bonus pts–see 'bonuses and incentives' in web-page)** Consider the following unusual approach to sorting, where the complexity is calculated based on a "flip" or exchange of 2 elements. (Here you don't care about how many comparisons the algorithm needs).

   - Show that sorting n elements can always be done with $O(n)$ flips in the worst case. Assume the elements are arranged in a doubly linked list, or array, whichever is more convenient for you.
   - Can a sorting algorithm work with simultaneous bounds of $O(n)$ exchanges and $O(n \log n)$ comparisons? Explain.
   - A group of n people each has a distinct piece of gossip. Person A can acquire a piece of gossip only directly from another person, say B, using 1 "whisper" in which B talks and A listens. Any whisper can include any number of pieces of gossip, but the listener and the speaker are fixed (they cannot even interchange roles during a whisper), and a whisper cannot be interrupted by other whispers. Give the best algorithm you can which uses whispers as basic operations to get each of the n people to possess all n pieces of gossip. You can assume the people are all arranged in an array or doubly-linked list or whatever is convenient.