CS311:	**Homework 3, Modified Friday afternoon, Sep. 19, 2008**	Due in class Tuesday, Sep. 23, 2008

Problems:

1. [25 pts.] Let $f, g, h : \mathbf{Z}^+ \rightarrow \mathbf{R}^+$, and let $c \in \mathbf{R}$.

   (a) In the discussion section we solved part (a) namely we defined:

   $$\lim_{n\to\infty}(h(n)) = c \quad \equiv \quad \forall \epsilon > 0\ \exists N \in \mathbf{Z}^+\ \forall n \geq N\ (|h(n) - c| < \epsilon)$$

   (b) Prove that if $\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = 0$, then $f = O(g)$, but $g \neq O(f)$.

   (c) Optionally, for some extra credit, prove or disprove the converse, i.e., must it be the case that if $f = O(g)$, but $g \neq O(f)$, then $\lim_{n\to\infty}\left(\frac{f(n)}{g(n)}\right) = 0$?

2. [25 pts.] Solve the following recurrence equations and express your answer using big oh notiation, e.g., $f = O(g)$.

   (a) $T_a(n) = 2T_a(n/3) + 3$

   (b) $T_b(n) = 9T_b(n/3) + 3n^2$

   (c) $T_c(n) = 2T_c(n/3) + n^2$

   (d) $T_d(n) = 2T_d(n-1) + n^2$

   Note: In the remainder of this course, as in the next two problems, you will frequently see the instructions, "Design and analyze an efficient algorithm for the following problem." What this means is the following. I want you to give a **clear** description of an algorithm in some combination of English and pseudo code. If your algorithm uses another algorithm that we have already seen in class or in the reading, then you may quote the name of that algorithm and its running time. You do not have to write the pseudo-code for that previously studied algorithm. Then, you should clearly analyze the running time of your algorithm, showing your answer using big Oh notation. Unless I say otherwise, you should analyze the worst case complexity. For such problems it is useful to first look at a few simple examples to get an idea of what an efficient algorithm would be.

   What we will look for in grading your solutions to such algorithms is the following:

   1. A clearly described, correct algorithm.

   2. A correctly analyzed algorithm that is as efficient as possible, i.e., if it was possible to do $O(n)$ then I will not be happy with $O(n \log n)$ and I will be very unhappy with $O(n^4)$.

   3. A brief, clear explanation of why your algorithm gives the correct answer on all inputs.

3. [25 pts.] Suppose that you are given as input d linked lists of long-integer elements such that each list is sorted. Your output should be a single, sorted linked list containing all the elements from the d input lists. Describe and analyze an efficient algorithm that merges these d lists into one sorted list in time $O(n \log d)$ where n is the sum of the lengths of the d lists. Note that d is a variable. It is part of the input, not a constant.

4. [25 pts.] You are given n records of the form: (customer's name, item purchased, price). Design and analyze an efficient algorithm to generate a list of those customers who have spent a total of at least \$1,000.

   For example, an input of the records:

   (Smith, TV, 300); (Jones, Car, 5000); (Black, Bicycle, 250); (Smith, Freezer, 700); (Brown, Calculator, 25),

   would result in the output: "Jones, Smith". (I don't care in what order the names are printed out.)