22C:31 Exam 1

Due via ICON dropbox by 5:30 pm, Friday, Feb 19th

You should feel free to use your textbook and notes. However, you should not seek help on the internet or from any other person. By turning in your exam, you are confirming that you've completed your exam within these constraints. You may send me e-mail seeking clarifications; these I can provide, but I will not provide any hints. The exam is worth 150 points. This translates to 15% of your grade.

1. **45 points** Here is a problem on running time analysis.

   (a) **20 points.** Compare the following pairs of functions in terms of how fast they grow asymptotically. In each case, say whether $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$, or $f(n) = \Theta(g(n))$. You don't have to provide any justification for your answers.
   Notes: (i) When $f(n) = \Theta(g(n))$, of course it is also true that $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$, but in this case the correct answer is $f(n) = \Theta(g(n))$ because this is the most specific. (ii) $n! = 1 \times 2 \times \cdots \times (n-1) \times n$.

	$f(n)$	$g(n)$
a.	$100n + \log n$	$n + 800(\log n)^2$
b.	$\log \sqrt{n}$	$\sqrt{\log(n)}$
c.	$\frac{n^2}{\log n}$	$n^2 - (\log n)^2$
d.	$(\log n)^{\log n}$	$\frac{n}{\log n}$
e.	$\log(n!)$	$n \log n$
f.	$n2^n$	$(1.5)^n$

   (b) **25 points** Consider the following pseudocode.

   ```
   while(n >= 0)do
     m ← n
     while(m >= 1)do
       m ← m − 1
     n ← n/2
   ```

   Express the running time of this code fragment as a function of n. Specifically, come up with a function $f(n)$ such that the running time of the function is $\Theta(f(n))$. Prove your claim by showing that the running time of the code fragment is both $O(f(n))$ and $\Omega(f(n))$.

2. **45 points.** In the Minimum Lateness problem our goal is to minimize the maximum lateness of any job. Consider a variant of this problem in which our goal is to minimize the sum of the lateness of all of the jobs. Let us call this the Minimum Sum Lateness problem.

   (a) **20 points** Consider the algorithm that schedules jobs in non-decreasing order of deadlines. Does this algorithm always produce an optimal solution to Minimum Sum Lateness? Justify your answer by either presenting a counterexample of presenting a proof of correctness.

   (b) **25 points** Consider the algorithm that schedules jobs in non-decreasing order of *slack*, i.e., $d_i - t_i$. In other words, compute for each job i a slack $s_i = d_i - t_i$ (note: s_i can be negative). Then order the jobs in non-decreasing order of the s_i-values. Does this algorithm always produce an optimal solution to Minimum Sum Lateness? Justify your answer by either presenting a counterexample of presenting a proof of correctness.

3. **60 points** Design a greedy algorithm for the *fractional knapsack problem* (see Homework 2), that always produces an optimal solution. Present the algorithm (in pseudocode) and present its proof of correctness.

   I realize that this problem also appears in Homework 2. I give this problem in the hope that you've already thought about it.
