Name:

Score:

1	2	3	4	5	6	7	SUM
10	15	15	10	20	15	15	[100]

CRN 21546 CIS 621 Winter 2009

Final Exam

(due by 10:15 am. on Wednesday, March 18)

This is the usual *open-everything, but no outside help take-home* test. Check "Class News", where I will post "frequently asked questions" about the test. Make sure that your answers are neat (preferably one problem per page) and legible. Please include this cover page with your submission.

1. **Loop Invariant** Given an array of "coefficients" *A[0..n]* and a value x, what is the value of *poly* that the following algorithm computes?

   i:=n; poly:=0; **while** $i \geq 0$ **do**
   begin *poly:=A[i]+x·poly; i:=i–1* **end**;

   Prove your answer by stating a loop invariant Inv and arguing that
   (i) the initialization establishes Inv;
   (ii) Inv is maintained by a single execution of the body of the loop;
   (iii) upon exit from the loop, *poly* holds the postulated value.

2. **Amortized Complexity** A mergeable priority queue of n elements can be implemented by lazy binomial heaps with constant time *Merge* operation.

   (i) Argue that infrequent ($m \in \omega(n)$) *DeleteMin* operations can lead to faster performance of the priority queue than what is impplied by the lower bound complexity of the comparison-based sorting problem.

   (ii) What should m be for a constant time amortized complexity of this implementation?

   (iii) Marking an element *deleted* in constant time implements a "lazy" general delete in a priority queue. Show how this can improve the amortized performance of the data structure when *DeleteMin* operations are infrequent.

3. **Greedy Algorithms** To merge two ordered files of sizes p and q, we move all their $p + q$ elements to a new, ordered file. Assume that a set of n ordered files, $m_1, m_2, ..., m_n$ is given together with their sizes $s_1, s_2, ..., s_n$. The problem is how to create their sorted union through $n-1$ merges so as to minimize the number of moved elements. Give a linear ($\mathcal{O}(n)$ time) algorithm to optimally *schedule* the merges in the case when the files are given in order of their sizes, $s_1 \leq s_2 \leq, ..., \leq s_n$.

4. **Greedy Algorithms vs. DP** Both the prefix-free code problem ("Huffman's code") and the OBST problem (with data in the leaves) strive to minimize the weighted external path length in a tree. Why does a greedy algorithm work for the former but not for the latter?

5. **Dynamic Programming** Consider an n-gon on the plane (each of the n vertices is given by a pair of coordinates) that is convex (the straight line joining any two interior points does not intersect its sides). A triangulation of the n-gon includes $n-3$ diagonals that divide its interior into $n-2$ triangular regions; the weight of a triangulation is the total length of its diagonals.

   Design an efficient algorithm finding a triangulation of the minimum total weight.

6. **Polynomial-time reductions** A language L is *complete* for a language class C with respect to polynomial-time reduction $\leq_p$ if $L \in C$ and for all $L' \in C, L' \leq_p L$.

   (i) Show that $\emptyset$ and $\{0,1\}^*$ (the set of all strings) are the only languages in P that are not P-complete with respect to polynomial-time reductions.

   (ii) Prove that the relation $\leq_p$ is transitive and reflexive. Is it symmetric?

7. **NP-completeness** Assume that there is a polynomial time algorithm CLQ to solve the *MaximumClique* decision problem:
   Instance: graph G and integer K
   Question: Does G have a completely connected set of K vertices?

   (i) Show how to use CLQ to determine the maximum clique **size** of a given graph in polynomial time.

   (ii) Show how to use CLQ to f**ind** a maximum clique of a given graph in polynomial time.

   (iii) Show that CLQ is NP-complete.

Extra Credit Describe a linear-time algorithm for the following problem: Given n distinct numbers $a_1, a_2, \ldots, a_n$ to which are assigned positive weights $w(a_1), \ldots, w(a_n)$, and a real number r, $0 < r \leq \sum_{1 \leq i \leq n} w(a_i)$. Determine m such that

$$\sum_{a_i < a_m} w(a_i) \;<\; r \;\leq\; \sum_{a_i \leq a_m} w(a_i).$$

(Hint: Interpret the problem when $w(a_i) = 1$, for all i, and $r = n/2$.)