You have three hours to work on this exam. You should return the exam to my office within 20 minutes of your finish time. The exam should be submitted no later than 11:59 PM on Saturday, October 6, 2001.

You may use a one page cheat sheet to help you on this exam. You may not consult any other texts or notes during this exam. You may not discuss the questions with anyone. You may not use a calculator during the exam.

Your answers will be judged for clarity, conciseness, and correctness. Algorithms and bounds should be, respectively, as fast and as tight as possible. The space provided should be adequate for your solutions.

The exam consists of five problems and 11 pages. Before you begin, make sure your exam is complete. Then write your name and start time below. When you complete the exam fill in your stop time.

Name ____

I started this exam at __________________________

I finished this exam at __________________________

1. (10 points) Order the following functions by increasing rate of growth. Identify any equivalence classes you find.

$$\log_{10} 3n, \;\; n \lg n, \;\; \ln^2 n, \;\; \lg\lceil\lceil n\rceil\rceil, \;\; \lg^{\lg n} n, \;\; 2^{\lg n}$$

2. (10 points) State whether each of the following are True or False. You should assume that the functions $f(n), g(n)$ and $h(n)$ referred to in these problems are monotonically increasing functions mapping the natural numbers to the natural numbers.

   (a) For every input size n, there is some input for which algorithm $\mathcal{A}$ takes exactly $2n^3$ steps. Thus we can conclude that the running time of $\mathcal{A}$ is $O(n^3)$.

   (b) If $f(n) = O(g(n))$ then $\lg(f(n)) = O(\lg(g(n)))$.

   (c) If $f(n) = O(g(n))$ and $g(n) = O(h(n))$ then $f(n) = O(h(n))$.

   (d) If $f(n) \neq O(g(n))$ then $f(n) = \Omega(g(n))$.

   (e) If $f(n)$ is polynomially bounded then $f(n) = O(2^n)$.

3. (20 points) Following are the recursive definitions of five functions:

f_1 and f_2: $f_1(1) = f_2(1) = 1$ and for $n > 1$

$$\begin{array}{lll} \text{for } n \text{ even:} & f_1(n) = f_1(n-1) \quad \text{and} & f_2(n) = 2^{f_1(n-1)} \\ \text{for } n \text{ odd:} & f_1(n) = 2^{f_2(n-1)} \quad \text{and} & f_2(n) = f_2(n-1) \end{array}$$

f_3 and f_4: $f_3(1) = f_4(1) = 1$ and for $n > 1$

$$\begin{array}{lll} \text{for } n \text{ even:} & f_3(n) = f_3(n-1) \quad \text{and} & f_4(n) = 2f_3(n-1) \\ \text{for } n \text{ odd:} & f_3(n) = 2f_4(n-1) \quad \text{and} & f_4(n) = f_4(n-1) \end{array}$$

f_5: $f_5(1) = 1$ and for $n > 1$

$$\begin{array}{ll} \text{for } n = \lceil\lceil n \rceil\rceil : & f_5(n) = 2 * f_5(n-1) \\ \text{for } n \neq \lceil\lceil n \rceil\rceil : & f_5(n) = f_5(n-1) + 1 \end{array}$$

Compute the values of each of these functions for $n = 2, 3, 4, 5$. (Some values may be quite large; if you find it more convenient you may express your answer as a power of 2.)

i	$f_i(1)$	$f_i(2)$	$f_i(3)$	$f_i(4)$	$f_i(5)$
1	1				
2	1				
3	1				
4	1				
5	1				

Complete the following table with Y=yes and N=no indicating whether $f_i(n) = \Gamma(f_j(n))$ for the values of i and j shown, where Γ is $O, \Omega, \Theta, o, \omega$.

i	j	O	Ω	Θ	o	ω
1	2					
1	3					
1	5					
3	4					
3	5					

4. (30 points) Algorithm $\mathcal{A}$ is recursive. (On inputs of other sizes the algorithm simply halts.)

   On an input size of 1 or 2, the algorithm does a constant amount of work. For input size of $n > 2$, the algorithm makes three recursive calls; two calls are made on subproblems of size $\frac{n}{4}$ and the other call is on a subproblem of size $\frac{n}{2}$. $\mathcal{A}$ then does an additional cn steps to compute its output. In other words, the recurrence relation for the running time of $\mathcal{A}$ is $T(n) = 2T(n/4) + T(n/2) + cn$. You may assume the input size is a power of 2.

   (a) Draw the work tree for the cases when $n = 1, 2, 4$ and 8. (You should notice that some root-to-leaf paths in the work tree are shorter than others.)

(b) Consider the work tree for $n = 4^m$, where $m \geq 2$. Let P denote a shortest root-to-leaf path in this tree. Let R denote a longest path.

i. What are the input sizes associated with the nodes along P, starting at the root?

ii. Which level contains the leaf of P? (The root is at level 0.)

iii. What are the input sizes associated with the nodes along R, starting at the root?

iv. Which level contains the leaf of R?

(c) Repeat the previous question (all parts) when $n = 2 * 4^m$, $m \geq 2$.

(d) Let ℓ denote the level containing the leaf of path P. Prove that for any level j, where $0 \leq j < \ell$, the total work done by the nodes at level j is cn.

(e) Using big-O notation, give the best upper bound you can for the running time of $\mathcal{A}$. Prove your answer.

(f) Using big-O notation, give the best lower bound you can for the running time of $\mathcal{A}$. Prove your answer.

5. (30 points) There is a party going on with n guests, $G_1, \ldots, G_n$. Every guest knows her/his own name. (The party has just begun.) A guest may also know the names of some other guests. If G_i knows the name of G_j, it need not be the case that G_j knows the name of G_i.

   A "celebrity" is someone whose name is known by all the guests but who does not know the name of anyone else. You are assigned the task of determining whether or not there is a celebrity at the party. To do this you are allowed to ask questions of the form "Does G_i know the name of G_j?"

   (a) Explain why there can be at most one celebrity at the party.

   (b) Give an algorithm for determining whether or not there is a celebrity at the party. To receive full credit your algorithm should ask as few questions, in worst case, as possible. How many queries, in worst case, does your algorithm require?

(c) Using an adversary argument, prove the best lower bound you can for the number of queries needed to solve this problem. (Note: A correct proof for a poor bound will garner more points that an incorrect proof for a good bound. In other words, only claim what you can prove.)