This is a long problem set. I encourage you to discuss the problems with each other, but remember that you must write up your solutions **independently**, and **list your collaborators** by name clearly at top of your submission. I take this disclosure very seriously.

74 points total. Note: whenever I say "show" I mean **prove**.

1. (4 pts) Suppose you have two algorithms to solve a particular problem, A taking time $f(n)$ and B taking time $g(n)$. Design an algorithm for this problem which achieves running time $O(\min(f(n), g(n)))$. Can you achieve running time exactly $\min(f(n), g(n))$?

2. (4 pts) Show (prove) that for any real constants a and b, where $b > 0$, the asymptotic relation $(n + a)^b = \Theta(n^b)$ is true.

3. (10 pts) Sort the following *functions* by order of asymptotic growth such that the final arrangement of functions $g_1, g_2, \ldots, g_{12}$ satisfies the ordering constraint $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, ..., $g_{11} = \Omega(g_{12})$. Give the final sorted list and identify which pair(s) of functions $f(n), g(n)$, if any, are in the same equivalence class, i.e., $f(n) = \Theta(g(n))$.

n	n^2	$(\sqrt{2})^{\lg n}$	$2^{\lg^* n}$	$n!$	$(\lg n)!$	$\left(\frac{3}{2}\right)^n$	$n^{1/\lg n}$	$n \lg n$	$\lg(n!)$	e^n	1

4. (6 pts total) Solve the following recurrence relations. Show all your work.

a. (3 pts) $T(n) = T(n-1) + n$, $T(1) = 1$

b. (3 pts) $T(n) = 2T(n/2) + n^3$, $T(1) = 1$ (assuming n is a power of 2)

5. (10 pts total) Roughly speaking, we say an algorithm is *data-aware* if it achieves a faster running time under some additional assumption which we expect to arise in practice. Here are two such examples for sorting an array.

a. (4 pts) Show that if an array is *nearly sorted*, in the sense that only k elements array are out of order for some constant k, then insertion sort runs in $O(n)$ time.

b. (4 pts) Suppose you are given a guarantee that every entry in an array is a positive integer, and at most some constant k. Give an $O(n)$ algorithm to sort such arrays.

c. (2 pts) Wait a minute, why doesn't (**b**) contradict the $\Omega(n \log n)$ lower bound given on page 59 of the textbook?

6. (14 pts total) The ever-despicable Gru has n minions, some of whom are *good* and always tell the truth, and some which are *bad* and unreliable. Gru constructs a test chamber in the hopes of determining which minions are good. The chamber holds two minions at a time, and when loaded, each minion sizes up the other and reports whether the other is good or bad. A good minion always reports accurately whether the other minion is good or bad, but the answer of a bad minion cannot be trusted. Thus, the four possible outcomes of a test are as follows:

Minion A says	Minion B says	Conclusion
B is good	A is good	both are good, or both are bad
B is good	A is bad	at least one is bad
B is bad	A is good	at least one is bad
B is bad	A is bad	at least one is bad

a. (5 pts) Show (prove) that if $n/2$ or more minions are bad, Gru cannot necessarily determine which minions are good using any strategy based on this kind of pairwise test. Assume that the bad minions can conspire to fool Gru.

b. (5 pts) Consider the problem of finding a single good minion, assuming that more than $n/2$ of the minions are good. Show (prove) that $\lfloor n/2 \rfloor$ pairwise tests are sufficient to reduce the problem to one of nearly half the size.

c. (4 pts) Show (prove) that the good minions can be identified with $\Theta(n)$ pairwise tests, assuming that more than $n/2$ of the minions are good.

7. (10 pts) In the game of Scrabble or Bananagrams, one is frequently faced with the following problem: given n tiles which are all face-up, flip them all to be face-down. Naturally one could simply flip the tiles one-by-one, but clearly this takes $O(n)$ time. Can we do better? Suppose you realize (as I did!) that you could save time by gathering some of the tiles into a box and then spilling them back out onto the table, thus flipping about half of them on average. Of course, gathering the tiles takes time as well, time which is roughly inversely proportional to the number of tiles being gathered.

Formally, assume that at any time you can perform one of the following actions:

a. Flip a single tile of your choice, taking one time step;

b. Gather k tiles of your choice in time n/k and spill them onto the table, flipping each with probability $1/2$. (Note that this could actually flip some tiles from face-down to face-up!)

Design and analyze an algorithm which flips all tiles face-down in *expected* time $o(n)$.

8. (8 pts) Graph problem: 4.14

9. (8 pts) Graph problem: 4.19

10. Optional (hard) problem: (0 pts + fame, glory, and maybe lunch)

(Based on a question from an actual Google software engineering interview.) Consider the following algorithm, which takes as input two integers n and k: set $x_0 = n/2$, and for each iteration t from 1 through k, set $x_t = (x_{t-1} + n/x_{t-1})/2$, finally outputting x_k.

a. What does x_k converge to as $k \to \infty$? (Prove it.)

b. Show how to view this algorithm as a binary search.

c. Let x^* be the answer from (**a**). Given some error $\epsilon > 0$, how many iterations does it take (i.e. what is k) until $|x^* - x_k| < \epsilon$? (Prove it.)

d. What happens when we change the update $x_t = (x_{t-1} + n/x_{t-1}^b)/2$ for $b = 2$ and $b = 3$? (Plots will suffice for evidence.)