CS 252: Algorithms
Exam 2. Due on paper by 8:30AM Monday, 9 November 2015.

This is an open-book, open notes, open-Internet exam. You may not consult with people other than Jeff Ondich. Use LaTeX as much as you can, but you may include neatly hand-drawn diagrams if appropriate.

1. (6 points) **Yet another Ω problem**. Let $f(N) = N^2 - 23N + 4$. Show that $f(N) \in \Omega(N^2)$. Your argument should have two steps:

   (a) Your first sentence should be "Let $c =$ *something* and $N_0 =$ *something*." I don't want to see your scratch work through which you determine c and N_0. Just start be saying what they are.

   (b) The rest of your argument should flow from "Suppose $N \geq N_0$..." and conclude...well, what it should conclude is described by the definition of Ω, and that's up to you to ensure you know.

   Yes, I'm asking you once again to do a simple complexity argument from the definitions. My goal on this problem is for all of you to be able to do this one perfectly.

2. (8 points) **Quicksort pivots**. When you implement *quicksort*, your key decision is how to select your *pivot*. (Don't recall what those words mean? Go find an introduction to quicksort.)

   (a) If you pick the first item in your array as your pivot, what's the worst case behavior of quicksort? Explain why using a recurrence relation argument.

   (b) Suppose you could find the median of your array in $O(1)$ time and then use the median as your pivot. What would the complexity of the worst case of quicksort be? Again, explain in detail.

   (c) OK, so you can't find the median $O(1)$ time. But you *can* find the median in $O(N)$ time, and then use it as your pivot. How would this affect the worst-case complexity of quicksort? Explain.

3. (10 points) **Stable dance partners**. To reduce the stakes of our stable marriage scenario, we're going to imagine that we have a set $E = e_1, e_2, ..., e_n$ of extroverts and a set $I = i_1, i_2, ..., i_n$ of introverts. Every extrovert wants to dance with an introvert, and vice versa. As in our previous discussions, the Gale-Shapley algorithm will begin with e_1 asking his/her top-ranked introvert to dance, followed by a response of "maybe" from the introvert in question, and so on. We'll also use the term *matching* to mean a list of (e_j, i_k) pairs, where each e_j and each i_k appears in exactly one pair–that is, everybody is dancing with exactly one partner, and each partnership has one extrovert and one introvert.

   (a) Suppose you measure the *total sadness* of a matching as the sum of all the ranks of the partners in the matching. (For example, if e_1 is dancing with her 2nd-favorite introvert i_7, and e_1 is i_7's 6th-favorite extrovert, then the pairing (e_1, i_7) will contribute 8 to the matching's total sadness.) Does a stable matching necessarily minimize sadness? Justify your answer via either a proof or a counter-example.

   (b) How many matchings are there? (This will give you an estimate on the running time of a brute-force approach to the search for a stable matching.)

   (c) Propose an efficient algorithm for testing a matching between E and I for stability. The inputs to your algorithm would be the preference list for each element of E or I, plus the matching itself. Analyze your algorithm's running time.

   (d) Could Gale-Shapley's running time be improved by using your stability-detection algorithm? Explain.

4. (3 points) I just finished a book, and I haven't decided what to read next. Any suggestions?

5. (8 points) **Something new**. Read about the *Boyer-Moore algorithm.* Its Wikipedia page is pretty good, but it also shows up in lots of Algorithms textbooks, and of course elsewhere on-line.

   Consider an instance of the Boyer-Moore algorithm where we are looking for the string *panamanian* inside the string *the man on an island with a mania for panama hats and anions is an amanuensis and is also panamanian.*

   (a) Show the data structures or tables that are computed during the preprocessing phase of the Boyer-Moore algorithm in this case.

   (b) List the character comparisons that are performed during the string matching phase of the algorithm. You'll need to come up with a way to express this clearly–perhaps lining up "panamanian" underneath the larger string at various horizontal positions will help.

   (c) How many character comparisons are made by the time the search string is found?

   (d) Very briefly, why is this algorithm relevant in bioinformatics?