This is an open-book, open-notes, open-internet exam. You may not speak with anyone other than me (Jeff Ondich) and Lily Haas about about this exam. You may, of course, ask questions in class and office hours. Cite your sources, explain things briefly but clearly, and make sure your submissions are readable (e.g., as LaTeXor clear images of handwritten text). Submit your work via Moodle. Have fun!

1. **A few important friends** [2.5]

   For each of the following, name an algorithm whose worst-case running time $T(n)$ satisfies the given constraint. (This is intended to be a very easy problem, but also a reminder that you should keep these examples in your head. The differences between these five complexity classes are an essential part of the day-to-day life of programmers everywhere, whether they know it or not.)

   (a) $T(n) = O(1)$

   (b) $T(n) = O(log\ n)$

   (c) $T(n) = O(n)$

   (d) $T(n) = O(n\ log\ n)$

   (e) $T(n) = O(n^2)$

2. **Common recurrences** [5]

   For each of the following, name an algorithm whose worst-case running time can be modeled by the recurrence relation in question. In addition, in each case give an $f(n)$ (without proof) for which $T(n) \in O(f(n))$. Wherever it appears, consider c to be a positive constant.

   (a) $T(n) = T(\frac{n}{2}) + c$

   (b) $T(n) = 2T(\frac{n}{2}) + cn$

   (c) $T(n) = T(n-1) + c$

   (d) $T(n) = T(n-1) + cn$

   (e) $T(n) = T(\frac{n}{5}) + T(\frac{7n}{10}) + cn$ (Hint: come to class Monday, Nov 14)

3. **Shipping schedules** [8]

   Due to slightly over-optimistic sales projections, I find myself with a basement full of boxes of Basque-Aymara/Aymara-Basque translation dictionary software for Amiga computers. Unable to bring myself to simply discard these lovingly crafted floppy disks, I am thrilled to discover that tomorrow, there is going to be an Andes/Pyrenees computer history conference in Ecuador. If I can just get these boxes to Quito by then, my software will support international understanding and good will.

I have a schedule F of n flights available today. Each flight F_k in this schedule is described by five values $F_k = (s_k, t_k, d_k, a_k, c_k)$:

s_k = starting city
t_k = ending city
d_k = departure time
a_k = arrival time
c_k = capacity (number of boxes of my software it can carry)

Describe an efficient algorithm for figuring out how to get as many boxes as possible from Minneapolis to Quito by tomorrow. You may assume that all the flights run on time, and that it takes no more than one hour to move the boxes from an arriving plane onto a departing plane. Make your algorithm as efficient as possible, and analyze its running time.

(You might find a perusal of the table of contents for Chapter 7 useful in getting started on this one.)

4. **Let's learn a new algorithm** [8]

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

$$S = \text{“panamanian”}$$

inside the string

$$T = \text{“the man on an island with a mania for panama hats and anions is an amanuensis and is also panamanian”}$$

(a) State, briefly, the purpose of the Boyer-Moore algorithm. Make clear what the expected inputs and outputs of the algorithm are.

(b) Show the data structures or tables that are computed during the preprocessing phase of the Boyer-Moore algorithm for the string S.

(c) List the character comparisons that are performed during the string matching phase of the algorithm as applied to the test case S and T. To make your exposition clear, you will need to come up with a way to express these character comparisons clearly (e.g., a diagram lining up “panamanian” beneath T at various horizontal positions or something like that).

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

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

5. **Help me have fun over break** [2]

   Over break, I plan to read and watch TV and movies and listen to music. What do you recommend? (You'll get the points whether you recommend something or not, but I do love getting recommendations from you.)

6. **A little dynamic programming** [8]

   Suppose you have a matrix of integers $M[1 \ldots w, 1 \ldots h]$ (where w is the width of the matrix and h is the height). A *path* through the matrix consists of a sequence of positions in the matrix, where there are three legal ways to move from one position to another: southwest (i.e. down one row and left one column), south (down one row and no change in column), and southeast (down one row and right one column). The *cost* of a path is the sum of the integers the path passes through.

   Your goal is to compute the lowest cost path that goes from the top row (in any column) to the bottom row (in any column). For example, for this matrix:

$$\begin{bmatrix} 1 & 2 & 0 & 4 \\ 0 & 1 & 3 & 3 \\ 5 & 5 & 1 & 6 \end{bmatrix}$$

   the cheapest path from top to bottom has value $2 = 0 + 1 + 1$.

   Let $S[x, y]$ = the cost of the cheapest path from position (x, y) in the matrix to any position in the bottom row.

   (a) Imagine a brute force solution that computes the cost of every possible path from the top row to the bottom row. Give a tight lower bound for the number of different paths in an $N \times N$ matrix. (Getting an exact count of paths near the left and right edges of the matrix is a little tricky, so just give an Ω bound here.)

   (b) Provide a recursive expansion of $S[x, y]$.

   (c) Describe the base case of your recursive expansion of $S[x, y]$.

   (d) In terms of S, how would you compute the cheapest path from top to bottom of a matrix?

   (e) Fill in a dynamic-programming-style chart for $S[x, y]$ and use it to show that your algorithm applied to this simple example matrix produces the correct answer 2.

7. **Have a great break!**

   It was a ton of fun working with you all this term. See you around.