This exam begins for you when you open (or peek inside) this packet. It ends at 12:00 noon on Monday 2012 May 14. Between those two times, you may work on it as much as you like. I recommend that you get started early and work often. The exam is open-book and open-note, which means, precisely:

(A) You may freely consult all of this course's material: the Sipser textbook, your class notes, your old homework and exam, and the materials on the course web site. If you missed a class and need to copy someone else's notes, do so before either of you begins the exam.

(B) You may assume all theorems discussed in class or in the assigned sections of the book. You do not have to prove or reprove them on this exam. On the other hand, you may not cite theorems that we have not studied. If you are unsure of whether you are allowed to cite a theorem, just ask.

(C) You may not consult any other papers, books, microfiche, film, video, audio recordings, Internet sites, etc. You may use a computer for these four purposes: viewing the course web site materials, editing and running Python programs relevant to the course, typing up your answers, and e-mailing with me. You may not share any materials with any other student.

(D) **You may not discuss the exam in any way (spoken, written, pantomime, semaphore, etc.) with anyone but me until everyone has handed in the exam — even if *you* finish earlier.** During the exam you will inevitably see your classmates around campus. Please refrain from asking even seemingly innocuous questions such as "Have you started the exam yet?" If a statement or question conveys any information, then it is not allowed; if it conveys no information, then you have no reason to make it.

During the exam you may ask me clarifying questions. If you believe that the statement of a problem is wrong, then you should certainly ask for clarification. Check your e-mail occasionally throughout the exam period, in case I send out a correction or clarification.

Your solutions should be thorough, self-explanatory, and polished (concise, neat, and well-written, employing complete sentences with punctuation). Always show enough work so that a classmate could follow your solutions. Do not show scratch work, false starts, circuitous reasoning, etc. If you cannot solve a problem, write a *brief* summary of the approaches you've tried. Submit your solutions in a single stapled packet, presented in the order they were assigned.

Partial credit is often awarded. Exam grades are loosely curved — by this I do not mean that there are predetermined numbers of As, Bs, Cs to be awarded, but rather that there are no predetermined scores required for grades A, B, C.

Good luck!

A. For any integer $n \geq 1$, let $b(n)$ denote its encoding in binary, with no leading 0s. For example, $b(12) = 1100$. Show that the language

$$A = \{b(n)\#b(n+1) : n \geq 1\} \subseteq \{0, 1, \#\}^*$$

is not context-free.

B. A (deterministic, single-tape) *two-dimensional Turing machine* is like our usual Turing machines, except that the tape is a two-dimensional plane of squares, like a checker board, extending infinitely in all four directions. When the machine starts up, its tape head is positioned over square $(0, 0)$, which contains a special symbol $\vdash \in \Gamma - \Sigma$, and the input string $w = w_1 w_2 \cdots w_n$ is in squares $(1, 0), (2, 0), \ldots, (n, 0)$. On each step of its computation, it moves either left, right, up, or down on the tape. Argue that two-dimensional Turing machines are equivalent to (deterministic, single-tape) ordinary Turing machines, by describing informally but clearly how each can simulate the other.

C. Let $DEC = \{\langle M \rangle : M \text{ is a Turing machine that halts on all inputs}\}$. Prove that DEC is neither recognizable nor co-recognizable.

D. Do not do this problem; hand in just the three other problems. In our interpreter assignment, we used a TreeNode class, in which each node object recorded its child nodes and its parent node. You were asked to write a scanner and parser. Suppose now that TreeNode does not record its parent node, but just its child nodes. How would your scanner and parser change? Be specific, so that I completely understand your algorithms. You may state your algorithms in Python (essentially redoing the assignment), in another common programming language (C, C++, Objective-C, Java, Scheme, etc.), or in precise English. (If, after solving this problem, you don't know why I've posed it, then you should carefully evaluate whether you've solved the problem.)