Your Name_____

Foundations of Computer Science
Exam #3

Lynn Andrea Stein
Franklin W. Olin College of Engineering

Your Name_____

This exam is intended as a three hour sit-down examination. However, you may self-administer it at any time between when it is distributed and 4pm on Friday December 17, 2004. Those who wish to take the exam during Friday's final exam period may do so; others do not need to attend the exam session on Friday 17 Dec, but must take responsibility for turning in their exams in AC318 by 4pm or in AC312 prior to that time. Regardless of when you complete the exam, you may not discuss it with anyone until after all exams have been turned in at 4pm on Friday Dec 17.

The exam is intended to be completed in a single sitting. You may take more than two hours to complete the examination, but you should not consider this an unlimited-time exam. (Taking four hours would be fine, though presumably unnecessary; taking 20 hours would not. Exercise reasonable judgement.) In particular, anything you can't solve within a reasonable amount of time is not likely to be worth an excessive effort.

This exam is closed book. You are not permitted to use any materials, or to consult with any people, beyond the exam itself or the course instructor. You may take reasonable breaks during the exam, but you are expected to honor the spirit of the single sitting administration. If possible, avoid mealtimes, conversations, phone calls, IMs, and other interpersonal interactions, though you may get up and walk around, have a snack, etc.

It is perfectly acceptable – even preferable – to hand write your answers in this exam booklet or on blank paper that you provide. If you wish to type your exam, you may use a computer but (a) you must not use resources on the computer other than your word processor (b) you should avoid checking email during the exam, except if that is your means of contacting me, and then only to read my email (c) you should not IM with people other than me.

Whether your exam is typed or hand written, each problem should be clearly identified, separated from other problems, and legible. Any extra pages should be stapled *in order* to the back of this exam and, on the problem page in this booklet, you should write "see attached page (#)." You may also continue solutions on the backs of pages or on additional pages, but these should also be clearly labeled and the exam book should note where the solution can be found.

After you have finished this exam, in the space provided on the final page or on an attached piece of paper, please write out the phrase "I have neither given nor received unauthorized assistance during the completion of this work. I agree not to discuss this exam in any way until after 4pm on December 17." Please sign your name to indicate that you have abided by all rules and conducted yourself according to the Olin College Honor Code. If you cannot write out this phrase and sign your name to it, please explain. [1]

[1] This text courtesy of Professor Sarah Spence.

1 Twenty Questions (Abbreviated Version)

This is a nonstandard multiple choice question. Circle the correct answer(s). In some cases, there may be more than one correct answer. The goal in this case is to circle all correct answers. Partial credit may be obtained by crossing out incorrect choices. Choices neither circled nor crossed out will be regarded as blank.

A. Which of the following operations is $\Theta(1)$?
   a. Insertion at the beginning of a linked list.
   b. Lookup of a key in a hashtable.
   c. Deletion from an array.
   d. Popping from a stack.

B. Which of the following lisp procedures is tail recursive?

a.
```
(define foo
  (lambda (x y)
    (if (= x 0)
        y
       (foo (- x 1) (+ y x))))))
```

b.
```
(define bar
  (lambda (x y)
    (if (= x y)
        0
       (+ x (bar (+ x 1) y))))))
```

c.
```
(define baz
  (lambda (x y)
    (if (= x y)
        x
       (+ y x (foo (+ x 1) (- y 1))))))
```

C. Computability theory tells us
   a. that it is impossible to build a program that takes another program as input.
   b. that it is impossible to build a program that takes another program as input and halts whenever its input program halts.
   c. that it is impossible to build a program that takes another program as input and halts whenever the input program fails to halt.
   d. that it is impossible to build a program that does not take any input.

D. Which of the following statements about order statistics is true?
   a. It is **not** possible to determine the largest element in a list of n elements in fewer than $n \log n$ comparisons.
   b. It **is** possible to determine the largest element in a list of n elements in fewer than n comparisons.
   c. It is **not** possible to determine the kth element in a list of n elements (for arbitrary $k < n$) in linear time.
   d. It **is** possible to determine the largest and smallest elements in a list of n elements in fewer than $(3n / 2) + 3$ comparisons.

E. The pumping lemma for regular expressions
   a. can be used to show that an expression is not regular.
   b. can be used to show that an expression must be context free.
   c. formalizes the intuition that if a string in the regular language is long enough, its finite state machine must loop.
   d. formalizes the intuition that finite state machines must halt.

F. For a list of length n
   a. Insertion sort is always the best sort to use.
   b. Merge sort must sort k lists of length n/k, for at least some values of $k \neq 1$.
   c. Selection sort looks at every unsorted element on each of its passes through the list.
   d. Bubble sort runs in time proportional to $n \log n$.

G. The value of the lisp expression `(car ''(x y z))` is
   a. `x`
   b. `y`
   c. `car`
   d. `quote`

H. First order predicate calculus (FOPC, known as Predicate Calculus to its friends)
   a. is incomplete (in the formal sense as used by Godel) because there are things that you cannot express in FOPC.
   b. is incomplete (in the formal sense as used by Godel) because there are things that you cannot prove in FOPC.
   c. is incomplete (in the formal sense as used by Godel) because you are not required to put a period at the end of each sentence.
   d. is not incomplete (in the formal sense as used by Godel).

I. Which of the following is true?
   a. Dijkstra's algorithm for shortest path is a greedy algorithm.
   b. Dijkstra's algorithm for shortest path is a divide and conquer algorithm.
   c. Dijkstra's algorithm for shortest path is a cdr-down-the-list algorithm.
   d. Dijkstra's algorithm for shortest path is an approximate algorithm.

J. The propositional formula **x (y)** is logically equivalent to
   a. **(x) y**
   b. **x (y)**
   c. **(x) y**
   d. **(y) x**

Your Name_____

2 Still More Fun With Turing Machines

A. Design a Turing Machine that takes as input a binary number and multiplies it by two. The format of the original TM tape is (for example):

…		+	1	0	0	0	1	1	1	0	1	1	.			…

That is, the head begins on a + that marks the first character of the input. (The head position is depicted as a black square.) The next n characters are 0s and 1s representing the binary number. After the final character of the binary number, the tape contains a decimal (well, binary) point, i.e., a period.

Your machine should end with the tape in the same format – a +, a binary number, and a . – but the binary number on the tape at the end of the computation should represent double the binary number with which you begin. The TM head should wind up on top of the plus sign at the end of the computation:

…		+	1	0	0	0	1	1	1	0	1	1	0	.		…

You may express your answer as a table of states and transitions or as a picture, whichever you find easiest. The answer should fit on the remainder of this page, although you are welcome to use additional space if you find that preferable. (Please be certain to put your name on any extra pages, to label them clearly and refer to them below, and, of course, to staple them into this package.)

B. Now design a Turing Machine that solves the same problem for decimal numbers. This TM takes as input a decimal number and multiplies it by two. The format of the original TM tape is (for example):

…		+	3	5	2	4	7	6	5	4	7	9	.			…

That is, the head begins on a + that marks the first character of the input. The next n characters are decimal digits representing the number. After the final character of the number, the tape contains a decimal point, i.e., a period.

Your machine should end with the tape in the same format – a +, a decimal number, and a . – but the number on the tape at the end of the computation should represent double the binary number with which you begin. The TM head should wind up on top of the plus sign at the end of the computation:

…		+	6	0	4	9	5	2	0	9	5	8	.			…

Again, you may express your answer as a table of states and transitions or as a picture, whichever you find easiest. The answer should fit on the remainder of this page, although you are welcome to use additional space if you find that preferable.

3 Automata / Chomsky Hierarchy

For each of the following languages, identify the **minimally powerful automaton** required to recognize it: a Finite State Machine (FSM), a Push-Down Automaton (PDA), or a Turing Machine (TM). In each case, only one answer is correct.

NOTE: You may cross out an answer that is incorrect to receive partial credit, i.e., you may earn points on this problem either by circling the correct answer or by crossing out the incorrect answers. Work shown will also be read for partial credit.

A. A particular language R that is regular.

FSM PDA TM

B. A particular language C that is context free.

FSM PDA TM

C. The language $\mathbf{a}^n\mathbf{b}^n$.

FSM PDA TM

D. The language $\mathbf{a}^n\mathbf{b}^n\mathbf{c}^n$.

FSM PDA TM

E. The language whose strings are restricted (lower case) variable names that are members of $\{\mathbf{a}, \mathbf{b}, \mathbf{c}, \ldots \mathbf{z}\}^+$

FSM PDA TM

F. The language of non-negative integers, i.e., the language whose strings are the members of $\mathbf{0} \cup \{\mathbf{0}, \mathbf{1}, \ldots \mathbf{9}\}\{\mathbf{0}, \mathbf{1}, \ldots \mathbf{9}\}^*$

FSM PDA TM

G. The language whose members are assignment statements similar to the following:

foo = 3

In this language, each string begins with a variable name that is a member of language E, followed by an **=** sign, followed by a non-negative integer, i.e., a member of the language F.

FSM PDA TM

H. The language whose members are multiple-assignment statements similar to the following:

[foo, bar, baz] = [3, 4, 10]

In this language, each string begins with a **[**, followed by one or more variable names (each a member of the language described in E), followed by a **]**, a **=**, and another **[**, followed by the *same number* of non-negative integers as there were variable names and a final **]**

FSM PDA TM

I. A simplified version of the language in which a variable must be declared before it is used. Specifically, if *var* and *type* are members of language E and *number* is a non-negative integer from language F, this language is all strings of the form:

var **:** *type*
var **=** *number*

Note that the two occurrences of *var* must be the *same* string from language E and that the strings in this language contain a newline (carriage return) in the middle, i.e., the two lines are both part of a single string.

FSM PDA TM

J. The language of properly nested, balanced parentheses, i.e., the language over the alphabet { (,) } in which every) has a preceding matching (. This language includes the following strings (among others):

(()) (()(()))() ()()(()())

FSM PDA TM

K. The language of properly nested, balanced parentheses, brackets, and braces, i.e., the language over the alphabet { (, {, [,), },] } in which every) has a preceding matching (. This language includes the following strings (among others):

([()]) [({}([]))()] ()[]({{}}()())

FSM PDA TM

4 NP-Completeness

The following two problem definitions are used in this question. However, only the optional bonus part of the question actually requires understanding these problems. For the required part of the problem, it suffices to know the names and inputs for these problems and to understand that 3-CNF-SAT is NP-complete.

> 3-CNF-SAT is the satisfiability problem for boolean formulas in 3-Conjunctive Normal Form. Specifically, if ϕ is a boolean formula in 3-CNF, ϕ is a conjunction of disjunctions of three propositions each:
>
> $$(x_{11} \quad x_{12} \quad x_{13}) \wedge (x_{21} \quad x_{22} \quad x_{23}) \wedge \ldots \wedge (x_{i1} \quad x_{i2} \quad x_{i3})$$
>
> **The 3-CNF-SAT Problem** is: Given as input a formula ϕ in 3-CNF, is there a truth assignment to the variables appearing in ϕ that makes the formula ϕ true?
>
> **The 0-1 Integer Linear Programming Problem** is: Given as input an integer $m \times n$ matrix $\boldsymbol{A}$ and an integer m-vector $\boldsymbol{b}$, is there an integer n-vector x with elements in the set $\{0,1\}$ such that $\boldsymbol{A}x \geq \boldsymbol{b}$?
>
> * * *

Suppose that you **already know that 3-CNF-SAT is NP-complete**. Which of the following things are necessary (and, as a group, sufficient) to **show that 0-1 Integer Linear Programming is NP-complete** (by means of a reduction involving 3-CNF-SAT)? Your answer should include only those things that are necessary **beyond** what is already given. [Note that you are not being asked to do the things listed here, just to identify which ones are necessary to make the argument go through.]

a. Give a polynomial time algorithm to find a solution to the 0-1 Integer Linear Programming Problem for arbitrary (i.e., general) input $\boldsymbol{A}$, $\boldsymbol{b}$.

NECESSARY NOT NECESSARY

b. Give a polynomial time algorithm to determine whether a particular n-vector x is a solution to the 0-1 Integer Linear Programming Problem for arbitrary (i.e., general) input $\boldsymbol{A}$, $\boldsymbol{b}$.

NECESSARY NOT NECESSARY

c. Give a polynomial time algorithm to find a solution to the 3-CNF-SAT Problem for arbitrary (i.e., general) ϕ.

NECESSARY NOT NECESSARY

d. Give a polynomial time algorithm to determine whether a particular truth assignment is a solution (i.e., a satisfying truth assignment) to the 3-CNF-SAT Problem for arbitrary ϕ.

NECESSARY NOT NECESSARY

e. Give a general algorithm for transforming a boolean formula ϕ into an integer $m \times n$ matrix $\boldsymbol{A}$ and an integer m-vector $\boldsymbol{b}$ …

NECESSARY NOT NECESSARY

(e.1) … with the property that ϕ has a satisfying truth assignment whenever there is an integer n-vector x with elements in the set $\{0,1\}$ such that $\boldsymbol{A}x \geq \boldsymbol{b}$

NECESSARY NOT NECESSARY

(e.2) …with the property that there is an integer n-vector x with elements in the set $\{0,1\}$ such that $\boldsymbol{A}x \geq \boldsymbol{b}$ whenever ϕ has a satisfying truth assignment

NECESSARY NOT NECESSARY

(e.3) …the algorithm for transforming ϕ into $\boldsymbol{A}$ and $\boldsymbol{b}$ must run in time **linear** in the size of ϕ

NECESSARY NOT NECESSARY

(e.4) …the algorithm for transforming ϕ into $\boldsymbol{A}$ and $\boldsymbol{b}$ must run in time **polynomial** in the size of ϕ

NECESSARY NOT NECESSARY

(e.5) …the algorithm for transforming ϕ into $\boldsymbol{A}$ and $\boldsymbol{b}$ must run **nondeterministically** in time **polynomial** in the size of ϕ

NECESSARY NOT NECESSARY

f. Give a general algorithm for transforming an integer $m \times n$ matrix $\boldsymbol{A}$ and an integer m-vector $\boldsymbol{b}$ into a boolean formula ϕ…

NECESSARY NOT NECESSARY

(f.1) … with the property that ϕ has a satisfying truth assignment whenever there is an integer n-vector x with elements in the set $\{0,1\}$ such that $\boldsymbol{A}x \geq \boldsymbol{b}$

NECESSARY NOT NECESSARY

(f.2) …with the property that there is an integer n-vector x with elements in the set {0,1} such that $\boldsymbol{A}x \geq \boldsymbol{b}$ whenever ϕ has a satisfying truth assignment

NECESSARY NOT NECESSARY

(f.3) …the algorithm for transforming $\boldsymbol{A}$ and $\boldsymbol{b}$ into ϕ must run in time **linear** in the size of A and b

NECESSARY NOT NECESSARY

(f.4) …the algorithm for transforming $\boldsymbol{A}$ and $\boldsymbol{b}$ into ϕ must run in time **polynomial** in the size of A and b

NECESSARY NOT NECESSARY

(f.5) …the algorithm for transforming $\boldsymbol{A}$ and $\boldsymbol{b}$ into ϕ must run **nondeterministically** in time **polynomial** in the size of $\boldsymbol{A}$ and $\boldsymbol{b}$

NECESSARY NOT NECESSARY

g. You may include any additional things that you would need to prove or demonstrate here or on a clearly marked additional page.

h. Which of the statements above would be necessary if you were trying to prove that show that 0-1 Integer Linear Programming is NP-**hard** rather than NP-complete? **Circle all that would be necessary; cross out all that would not be necessary.** Numbers not circled or crossed out will be considered blank.

a b c d e e.1 e.2 e.3 e.4 e.5

f f.1 f.2 f.3 f.4 f.5

g (specify)

Optional Bonus (Extra Credit Only):

On a separate piece of paper, clearly marked and with your name on it, prove that the 0-1 Integer Linear Programming Problem is NP-complete (by means of a reduction involving 3-CNF-SAT)

5 Design

This question is, to a first approximation, an essay question. You will want to answer it on separate paper (which you will dutifully and elegantly staple to this exam.)

For each part of this question, you should choose from among the mechanisms we have discussed this semester and outline a solution in terms of one or more of these mechanisms. Your answer may focus on any of the following (or other topics from this course) that may be appropriate:

- Stack, queue, hashtable, linked list, array, tree, graph.
- Finite state machine, push-down automaton, turing machine.
- Regular expression, context free grammar.
- Constant, logarithmic, linear, n log n, quadratic, exponential.

Where feasible, you should indicate the complexity class (e.g., using Θ notation) of your proposed solution.

Note that in this question you are *not* being asked to play the role of the programmer; you are the designer. Thus, your answer should not contain code unless you can find no other way to communicate your ideas. It should also not presume a particular programming language. Instead, your answer should be a description of the mechanisms that the programmer should use to implement your design.

This section is explicitly about making judgements, so the quality of your answer will depend as much on how you justify your selections as on what those selections are.

A. You are working on a project to help manage email. Your first task is to design something that will read an email message and extract the sender, date, and subject of the message. The structure of an email message looks like this:

From: Behavioral & Brain Sciences <calls@bbsonline.org>
Date: Tue Dec 14, 2004 8:07:06 PM US/Eastern
To: las@olin.edu
Subject: Ainslie/Breakdown of Will: BBS Multiple Book Review

The lines may not appear in this order. There will generally be other lines as well. Field lines will begin with a field name followed by a colon. A completely blank line marks the end of the fields and the beginning of the message body.

Recommend a mechanism to extract and record the three critical pieces of information from an email message. Include information about the expected running time of your approach.

B. Suppose now that you have a number of these email messages and you have extracted the same information from each of them. Describe a mechanism to store these messages so that you can retrieve them by date. Include information about why you have chosen this approach as well as about the expected running time of storage and of retrieval.

C. Oops! The boss changed the game plan. Now you need to be able to retrieve the emails by *any* of the three fields that you have extracted. Describe a new mechanism to store these messages *or* explain why the one you designed for part B is just fine even under the new requirements. Again, justify your approach and describe its properties, pros and cons.

D. Assume that you have the multi-way lookup mechanism that you've just described. Show how you could use it to determine whether you have received multiple messages from the same person on the same topic. (You may assume that these messages will contain the same subject headers, i.e., you do not need to deal with the presence of Re:, Fwd:, etc.)

E. Your information manager should not simply keep track of information that you've received, it should also present this information to you in a useful way. Assume that the number of emails you have received on a topic (not necessarily from the same person) is a good indicator of the importance of that topic. Describe a mechanism to present the emails (or at least their headers) in priority order. It may build on what you've already designed or it may be a different mechanism entirely. Again, explain why this is the mechanism you chose, its expected running time, costs and benefits, etc.

6 Honor Code Declaration

Please write out and sign the honor code declaration from the instructions on page 2 of this exam in the space below or provide an explanation here as to why you cannot do so.