Comprehensive Exam: Algorithms and Concrete Mathematics Autumn 2005

This is a one hour closed-book exam and the point total for all questions is 60.

In questions that ask you to provide an algorithm, please explain the algorithm in words and diagrams, no need to write code or pseudo code. Also, for any algorithm, state and prove its running time. No credit will be given for exponential-time algorithms. Polynomial but slow algorithms will get some partial credit. Amount of credit will depend on how much slower they are compared to what is achievable using the knowledge in the reading list.

For full credit, the answers should be short and precise. Long and convoluted answers will not get full credit even if they are correct.

The following is a statement of the Stanford University Honor Code:

A. *The Honor Code is an undertaking of the students, individually and collectively:*
   (1) *that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;*
   (2) *that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.*

B. *The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.*

C. *While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.*

By writing my "magic number" below, I certify that I acknowledge and accept the Honor Code.

_____

(Number)

Prob	# 1	# 2	# 3	# 4	Total
Score					
Max	10	20	15	15	60

1. **[10 pts]** Prove a tight asymptotic bound on the behavior of $T(n) = T(n-1) + \ln n$, where $T(1) = 1$.

2. **[20 pts]** Let $d > 0$ be a small integer. We study a heap structure where the heap branching factor is d rather than 2. We call such a heap a d-heap (the standard heap is a 2-heap). Note that a d-heap of depth ℓ has d^ℓ elements in it.

   a. Give an efficient algorithm for insert and extract-min for a d-heap. Give the asymptotic running time for your algorithm.

   b. Give the best algorithm you can think of for constructing a d-heap from a given vector of n elements (build-heap). Give the asymptotic running time for your algorithm.

   c. Describe a sorting algorithm that uses a d-heap. Give the asymptotic running time for your algorithm. The running time will depend on n (the size of the given list) and on d.

   d. What is the optimal value of d for getting the fastest sorting algorithm?

3. **[15 pts]** Let $G = (V, E)$ be a connected undirected graph. A bridge is an edge $e \in E$ such that removing e disconnects the graph, i.e. breaks the graph into at least two connected components. Give an $O(|E|)$ time algorithm to find all bridge edges of G.
   Hint: use DFS.

4. **[15 pts]** You are given an array of n objects. You are told that some element occurs at least $\lfloor n/2 \rfloor + 1$ times in the array. We call this element the majority element.

   Suppose the objects are totally ordered (that is you are given a function that takes two objects A and B as input and returns (key$(A) <$ key(B))). Give an algorithm that finds the majority element in time $\Theta(n)$.
   Hint: a one line answer is sufficient.

Stanford University
Computer Science Department

Fall 2005 Comprehensive Exam in Artificial Intelligence

1. **Closed Book - NO laptop. Write only in the Blue Book provided.**
2. **The exam is timed for one hour.**
3. **Write your Magic Number on this sheet and the Blue Book; DO NOT WRITE YOUR NAME.**

The following is a statement of the Stanford University Honor Code:

A. The Honor Code is an undertaking of the students, individually and collectively:
   1. that they will not give or receive aid in examinations; that they will not give or receive un-permitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
   2. that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

B. The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

C. While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

By writing my Magic Number below, I certify that I acknowledge and accept the Honor Code.

Magic Number----------

2005 Comprehensive Examination
Artificial Intelligence

1. Search. (20 points) Consider a search tree with uniform branching factor 2 and depth d, and consider a search problem for which there is a single solution in the tree at depth. Give expressions for the best and worst case cost of finding the solution, in terms of nodes visited, for (a) breadth-first search, (b) depth-first search, and (c) iterative deepening starting at depth 1 and incrementing by 1 on each iteration. For the purposes of this problem, you should assume that the root of the tree is at depth 1.

2. Logic. (20 points) Let Γ and Δ be sets of closed sentences in first-order logic, and let φ and ψ be individual closed sentences in first-order logic. State whether each of the following statements is true or false. No explanation is necessary.

(a) $\Gamma \models \varphi$ and $\Delta \models \psi$, $\Gamma \cup \Delta \models (\varphi \wedge \psi)$.
(b) $\Gamma \models \varphi$ and $\Delta \models \psi$, $\Gamma \cap \Delta \models (\varphi \vee \psi)$.
(c) $\Delta \models (\varphi \Rightarrow \psi)$ if and only if $\Delta \cup \{\varphi\} \models \psi$.
(d) $\Delta \models \varphi$ or $\Delta \models \psi$ if and only if $\Delta \models (\varphi \vee \psi)$.
(e) If $\Delta \models \varphi$ and $\Delta \models \psi$, then $\Delta \models (\varphi \Rightarrow \psi)$.
(f) If $\Delta \models p(\tau)$ for some ground term τ, then $\Delta \not\models \forall x.\neg p(x)$.
(g) If $\Delta \models p(\tau)$ for every ground term τ, then $\Delta \models \forall x.p(x)$.
(h) If $\Delta \models \exists x.(p(x) \Rightarrow q(x))$, then $\Delta \models \exists x.(p(x) \wedge q(x))$.
(i) If $\Delta \models \varphi$ and $\Delta \models (\varphi \Rightarrow \psi)$, then $\Delta \not\models \neg\psi$.
(j) If $\Gamma \models (\varphi \Rightarrow \psi)$ and $\Delta \models (\psi \Rightarrow \varphi)$, then $\Gamma \cap \Delta \models (\varphi \Rightarrow \psi) \vee (\psi \Rightarrow \varphi)$.

3. Automated Reasoning. (20 points) Use the resolution method and the following premises to prove the conclusion shown below.

Premises:

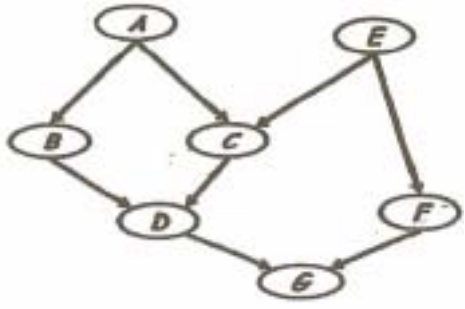
$\forall x.\forall y.\forall z.(p(x,y,z) \Rightarrow s(x,y,z))$
$\forall x.\forall y.\forall z.(q(x,y,z) \Rightarrow s(x,y,z))$
$\forall x.\exists y.\forall z.(p(x,y,z) \vee q(x,y,z) \vee \neg r(x))$

Conclusion:

$\forall x.(r(x) \Rightarrow \exists y.(s(x,y,b) \wedge s(x,y,c)))$

Note that this is a question about Resolution. You will get zero points (nil, nada, rien, zip, nothing) unless you prove it using the standard resolution procedure.

4. Bayes Nets. (20 points) Consider the following Bayesian.



(a) Write $p(G)$ as a sum of values from the full joint distribution for variables A,…, G. Equivalently, what is the mathematical expression computed in this case by the Enumeration-Joint-Ask algorithm described in Russell and Norvig?

(b) Write $p(G)$ as a nested sum of products of conditional probabilities from the tables associated with this Bayes Net. Equivalently, what is the mathematical expression computed in this case by the Enumeration-Ask algorithm described in Russell and Norvig?

(c) Assume we are trying to compute $p(G)$. Fill in the following table to show what the variable elimination algorithm will do on this case on the assumption that we are eliminating variables in the order B, C, D, F (with B being removed first, not last). Note: your generated factors should be written in the form $g_i(X_1, \ldots, X_l)$ that clarifies the variables involved in the factor. . Equivalently, write the factors that would be generated in this case by the Elimination-Ask algorithm described in Russell and Norvig.

Variable	Factors Used	Factors Generated
B		
C		
D		
F		

5. Learning. (20 points) Consider the following training set for a decision-tree learning problem. Here, a, b, c, d, and e are Boolean features, and $x_1, \ldots, x_6$ are samples.

Example	a	b	c	d	e	Goal
x_1	1	0	0	0	0	1
x_2	1	1	1	1	0	0
x_3	0	0	0	1	0	1
x_4	1	1	1	0	0	1
x_5	1	0	0	1	0	0
x_6	0	0	0	0	0	0

(a) Draw a decision tree of minimal depth that correctly classifies the examples in this dataset.

(b) How much information is needed to classify an example in this case? (Reminder: the amount of information needed to classify an example is $-(p*\log p)-(n*\log n)$, where p is the probability of a positive answer and n is the probability of a negative answer.)

(c) How much information is needed to classify an example *given* that a is 1? What if a is 0?

(d) So, what is the information gain from attribute a? What is the information gain from d given a?

2005 Comprehensive Examination Solutions
Artificial Intelligence

1. Search. (20 points) Costs for finding a solution at depth k in a tree with branching factor 2 with overall depth d.

Time	*Best*	*Worst*
Depth – First Search	k	$2^d - 2^{d-k+1} + 1$
Breadth – First Search	2^{k-1}	$2^k - 1$
Iterative Deepening	$2^k - 1$	$2^{k+1} - k - 2$

Iterative deepening is simply a sum of breadth-first search at levels up to k. A key benefit is that it uses only the same amount of space as depth-first search.

2. Logic. (20 points)
(a) True
(b) False
(c) True
(d) False
(e) True
(f) False
(g) False
(h) False
(i) False
(j) True

3. Resolution. (20 points)

Clauses:
1. $\{\neg p(x,y,z), s(x,y,z)\}$
2. $\{\neg q(x,y,z), s(x,y,z)\}$
3. $\{p(x,f(x),z), q(x,f(x),z), \neg r(x)\}$
4. $\{r(a)\}$
5. $\{\neg s(a,y,b), \neg s(a,y,c)\}$

Derivation:
6. $\{pa,f(a),z), q(a,f(a),z)\}$ 3,4
7. $\{s(a,f(a),z), q(a,f(a),z)\}$ 1,6
8. $\{s(a,f(a),z)\}$ 2,7
9. $\{\neg s(a,f(a),c)\}$ 5,8
10. $\{\}$ 8,9

4. Bayes Nets. (20 points)

(a) We compute $p(G)$ by summing over all cases in which the desired condition is true.

$$p(G)=\sum_{a}\sum_{b}\sum_{c}\sum_{d}\sum_{e}\sum_{f} p(a,b,c,d,e,f,G)$$

(b) Using the chain rule for probability, we can write $p(a,b,c,d,e,f,G)$ as follows.

$$p(a,b,c,d,e,f,G)=p(a)\,p(b|a)\,p(c|a,b)\,p(d|a,b,c)\,p(e|a,b,c,d)\,p(f|a,b,c,d,e)\,p(G|a,b,c,d,e,f)$$

Using conditional independences from the Bayes net, we can rewrite this more compactly.

$$p(a,b,c,d,e,f,G)=p(a)\,p(b\mid a)\,p(c\mid a,e)\,p(d\mid b,c)\,p(e)\,p(f\mid e)\,p(G\mid d,f)$$

Inserting this version into the expression from part (a), we get the following.

$$p(G)=\sum_{a}\sum_{b}\sum_{c}\sum_{d}\sum_{e}\sum_{f} p(a)p(b\mid a)p(c\mid a,e)p(d\mid b,c)p(e)p(f\mid e)p(G\mid d,f)$$

Finally, we can move some of the factors outside of sums, saving some multiplications.

$$p(G)=\sum_{a} p(a)\sum_{b} p(b\mid a)\sum_{c}\sum_{d} p(d\mid b,c)\sum_{e} p(c\mid a,e)p(e)\sum_{f} p(f\mid e)p(G\mid d,f)$$

(c)

Variable	Factors Used	Factor Generated
B	$p(b \mid a)\ p(d \mid b, c)$	$g_1(a, c, d)$
C	$p(c \mid a, e)\ g_1(a, c, d)$	$g_2(a, d, e)$
D	$p(G \mid d, f)\ g_2(a,d, e)$	$g_3(a, e, f,G)$
F	$p(f \mid e)\ g_3(a, e, f, G)$	$g(a, e,G)$

5. Learning. (20 points)

(a) The concept is *a xor d*; so a 2 level tree is all that is needed.

(b) 1 bit

(c) 1 bit in either case.

(d) 0 bits for a and 1 bit for d given a.

Automata and Formal Languages Comprehensive Exam (60 Points)

Fall 2005

Problem 1 (10 points)

(a) Show that if L is a regular language, then so is

$$min(L) = \{w : w \text{ is in } L, \text{ but no proper prefix of } w \text{ is in } L\}.$$

(b) Show that if L is a regular language, then so is

$$init(L) = \{w : \text{there exists an } x \text{ such that } wx \text{ is in } L\}.$$

Hint: For each part, start with a DFA for L and modify it. Include *brief* arguments justifying your constructions.

Problem 2 (15 points)

Decide whether the following statements are TRUE or FALSE. *You will receive 3 points for each correct answer and -2 points for each incorrect answer.*

(a) There is a language L such that neither L nor its complement are recursively enumerable.

(b) Suppose there is a polynomial-time reduction from the language L_1 to the language L_2. If L_1 is NP-hard, then L_2 must be NP-complete.

(c) Suppose there is a polynomial-time reduction from the language L_1 to the language L_2. If L_1 is NP-complete, then L_2 must be NP-hard.

(d) The following language is recursively enumerable: encodings of Turing machines that accept at least 154 different inputs.

(e) The following language is recursively enumerable: encodings of Turing machines that accept at most 154 different inputs.

Problem 3 (15 points)

Classify each of the following languages as being in one of the following classes of languages: *empty, finite, regular, context-free, recursive, recursively enumerable, all languages*. You must give the *smallest* class that contains *every possible language* fitting the following definitions. For example, the language of a DFA could be empty or finite, and must always be context-free, but the smallest class that contains all such languages is that of the *regular* languages. *You will receive 3 points for each correct answer and -2 points for each incorrect answer.*

(a) A subset of a regular language.

(b) The concatentation of two recursively enumerable languages. (Recall that the concatenation of languages L_1 and L_2 is $L_1L_2 = \{wx \mid w \in L_1, x \in L_2\}$.)

(c) The concatentation of two recursively enumerable languages, one of which is the complement of the other.

(d) An NP-complete language.

(e) An NP-hard language.

Problem 4 (20 points)

An instance of the Integer Linear Programming Problem is the following: given a set of linear constraints of the form $\sum_{i=1}^{n} a_i x_i \leq c$ or $\sum_{i=1}^{n} a_i x_i \geq c$, where the a's and c's are integer constants and x_1, x_2, ..., x_n are variables, does there exist an assignment of integers to each of the variables that makes all of the the constraints true? Prove that the Integer Linear Programming Problem is NP-hard.

Automata and Formal Languages Comprehensive Exam (60 Points)

Fall 2005

Problem 1 (10 points)

(a) Show that if L is a regular language, then so is

$$min(L) = \{w : w \text{ is in } L, \text{ but no proper prefix of } w \text{ is in } L\}.$$

(b) Show that if L is a regular language, then so is

$$init(L) = \{w : \text{there exists an } x \text{ such that } wx \text{ is in } L\}.$$

Hint: For each part, start with a DFA for L and modify it. Include *brief* arguments justifying your constructions.

Solution:

(a) Let M be a DFA accepting L (such a machine exists because L is regular). Obtain a new DFA M' from M as follows. Add an extra "null" state z which is non-accepting. All transitions from z lead back to z. To every accepting state q of M, add transitions (for all input symbols) from q to z. If $w \in L$ and no prefix of w is in L, then M will reach an accepting state for the first time once w is fully consumed. Thus w will lead M' to an accepting state. Conversely, if some prefix of w is in L, then M will first reach an accepting state before w is fully consumed. Thus on input w, the DFA M' will conclude in the null state. Thus M' accepts precisely the language $min(L)$.

(b) Let M be a DFA accepting L (such a machine exists because L is regular). Obtain a new DFA M' from M as follows. For every state q of M for which there is a (possibly empty) sequence of transitions from q to an accepting state of M, make q accepting in M'. (Other states remain non-accepting in M'.) Then an input w is accepted by M' if and only if there is a string x such wx is accepted by M. Thus M' accepts precisely the language $init(L)$.

Problem 2 (15 points)

Decide whether the following statements are TRUE or FALSE. *You will receive 3 points for each correct answer and -2 points for each incorrect answer.*

(a) There is a language L such that neither L nor its complement are recursively enumerable.

(b) Suppose there is a polynomial-time reduction from the language L_1 to the language L_2. If L_1 is NP-hard, then L_2 must be NP-complete.

(c) Suppose there is a polynomial-time reduction from the language L_1 to the language L_2. If L_1 is NP-complete, then L_2 must be NP-hard.

(d) The following language is recursively enumerable: encodings of Turing machines that accept at least 154 different inputs.

(e) The following language is recursively enumerable: encodings of Turing machines that accept at most 154 different inputs.

Solution:

(a) TRUE

(b) FALSE

(c) TRUE

(d) TRUE

(e) FALSE

Problem 3 (15 points)

Classify each of the following languages as being in one of the following classes of languages: *empty, finite, regular, context-free, recursive, recursively enumerable, all languages*. You must give the *smallest* class that contains *every possible language* fitting the following definitions. For example, the language of a DFA could be empty or finite, and must always be context-free, but the smallest class that contains all such languages is that of the *regular* languages. *You will receive 3 points for each correct answer and -2 points for each incorrect answer.*

(a) A subset of a regular language.

(b) The concatentation of two recursively enumerable languages. (Recall that the concatenation of languages L_1 and L_2 is $L_1L_2 = \{wx \mid w \in L_1, x \in L_2\}$.)

(c) The concatentation of two recursively enumerable languages, one of which is the complement of the other.

(d) An NP-complete language.

(e) An NP-hard language.

Solution:

(a) All languages

(b) Recursively enumerable

(c) Recursive

(d) Recursive

(e) All languages

Problem 4 (20 points)

An instance of the INTEGER LINEAR PROGRAMMING PROBLEM is the following: given a set of linear constraints of the form $\sum_{i=1}^{n} a_i x_i \leq c$ or $\sum_{i=1}^{n} a_i x_i \geq c$, where the a's and c's are integer constants and x_1, x_2, ..., x_n are variables, does there exist an assignment of integers to each of the variables that makes all of the the constraints true? Prove that the INTEGER LINEAR PROGRAMMING PROBLEM is NP-hard.

Solution: Recall the NP-complete 3-SAT problem: given a set of Boolean variables and a set of disjunctions of 3 literals each, is there a truth assignment that simultaneously satisfies all of the clauses? We can establish NP-hardness of ILP via a polynomial-time reduction from 3-SAT.

The reduction starts with a 3-SAT instance, given by variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$, where C_i has the form $l_1 \vee l_2 \vee l_3$, where l_1, l_2, l_3 are literals (each of the form x_j or $\neg x_j$ for some j). It then constructs an instance of ILP as follows. For each Boolean variable x_j, there is an integer variable y_j with

the interpretation that if $y_j = 1$ then x_j should be set to "true" and if $y_j = 0$ then x_j should be set to "false". For every j, add the constraints $x_j \geq 0$ and $x_j \leq 1$. For every j, introduce the variable z_j and the constraint $z_j = 1 - y_j$ to model the value of $\neg x_j$. (Strictly speaking, we accomplish this by adding the inequalities $z_j \geq 1 - y_j$ and $z_j \leq 1 - y_j$.) Finally, for every clause of the form $l_i \vee l_j \vee l_k$ we construct a constraint $(y_i, z_i) + (y_j, z_j) + (y_k, z_k) \geq 1$, where by (e.g.) (y_i, z_i) we mean y_i if $l_i = x_i$ and z_i if $l_i = \neg x_i$. This reduction runs in linear time, but it remains to verify its correctness.

First suppose that there is a satisfying truth assignment to the given 3-SAT instance. For each Boolean variable x_i, set the corresponding integer variable y_i to 1 if x_i is set to true in the satisfying assignment and to 0 otherwise. Set $z_i = 1 - x_i$ for each i. Each (y_i, z_i) equals 1 if and only if the corresponding literal is satisfied by the truth assignment. Since the truth assignment satisfies all of the 3-SAT clauses, the left-hand side $(y_i, z_i) + (y_j, z_j) + (y_k, z_k)$ of every constraint is at least 1. Thus the assignment to the the integer variables is also satisfying.

Conversely, suppose there is an assignment to the integral variables of the ILP instance that satisfies all of the linear constraints. Set x_i to true (false) if $y = 1$ ($y = 0$). Reversing the argument in the previous paragraph shows that this a satisfying truth assignment to the given 3-SAT instance.

Stanford University
Computer Science Department

Fall 2005 Comprehensive Exam in Compilers

1. **Closed Book, - NO laptop. Write only in the Blue Book provided.**
2. **The exam is timed for one hour.**
3. **Write your Magic Number on this sheet and the Blue Book; DO NOT WRITE YOUR NAME.**

The following is a statement of the Stanford University Honor Code:

A. The Honor Code is an undertaking of the students, individually and collectively:

1. that they will not give or receive aid in examinations; that they will not give or receive un-permitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;

2. that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

B. The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

C. While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

By writing my Magic Number below, I certify that I acknowledge and accept the Honor Code.

Magic Number----------

Compilers Comprehensive, November 2005

This is a 60 minute, closed book exam. Please mark your answers in the blue book.

1. (10 points)

   Suppose you were implementing a lexical analyzer for the C programming language in a tool like Lex or Flex. Suppose the entire input to the compiler were:

```
divbypointer(double num, double *pdenom)
{
  return num/*pdenom;
}
```

   Describe two ways of handling comments.

   (a) Method 1 uses a regular expression that matches a complete comment as a single lexeme.

   (b) Method 2 recognizes `/*` and then enters a special "start condition," in which `*/` and any single character are recognized as lexemes. When `*/` is recognized, it returns to the default start condition, in which C language tokens are recognized.

   Briefly discuss how each method would handle the example above, and discuss the practical merits of each.

2. (15 points) These questions are about the handling of variables by a compiler for a simple language like C. In your answers, address only the compiler behavior that is *necessary for code generation.* Do not address type checking or other aspects of semantic analysis are not strictly necessary to emit code for correct programs.

   (a) Describe the compiler's processing of a global variable g of type `int`, both at the point of *declaration* and at the point of *use*.

   (b) How is the handling of a local variable declaration of type `int` different from the handling of the global variable of the same type?

   (c) What information does the compiler have to maintain in the symbol table to generate code for `A[i].f[j]`?

3. (10 points)

   Why would it be useful for an optimizing compiler to have optimizations on an intermediate representation (such as 3-address code) *and* peephole optimization at the instruction level?

4. (35 points) Consider the following context-free grammar:

$$
\begin{aligned}
S &\rightarrow Sa \\
S &\rightarrow bS \\
S &\rightarrow c
\end{aligned}
$$

(a) (3 points) Show that the grammar is ambiguous.

(b) (10 points) Write the canonical collections of LR(1) items for this grammar.

(c) (2 points) Identify all conflicting items, and the types of the conflicts (e.g., "shift-reduce conflict in state 3 on d").

(d) (5 points) Could the original grammar be converted into an LALR(1) parser that parses all input correctly by resolving conflicts, in the way that YACC and similar parser generators allow? If so, how should they be resolved? In either case, please explain (briefly).

(e) (5 points) Rewrite the grammar in an equivalent form that is suitable for LL parsing and minimizes the use of stack space.

(f) (5 points) Rewrite the grammar in an equivalent form that is directly suitable for LR parsing (i.e., does not result in conflicts) and minimizes the use of stack space.

(g) (5 points) In your modified LL(1) grammar, show the sequence of stack contents and inputs when parsing the input *bbcaa*.

Solutions to Compilers Comprehensive, November 2005

This is a 60 minute, closed book exam. Please mark your answers in the blue book.

1. (10 points)

   Suppose you were implementing a lexical analyzer for the C programming language in a tool like Lex or Flex. Suppose the entire input to the compiler were:

```
divbypointer(double num, double *pdenom)
{
  return num/*pdenom;
}
```

   Describe two ways of handling comments.

   (a) Method 1 uses a regular expression that matches a complete comment as a single lexeme.

   (b) Method 2 recognizes `/*` and then enters a special "start condition," in which `*/` and any single character are recognized as lexemes. When `*/` is recognized, it returns to the default start condition, in which C language tokens are recognized.

   Briefly discuss how each method would handle the example above, and discuss the practical merits of each.

   Solution:

   Method 1 would compile the above program without errors. There are many disadvantages to method 1. The regular expression is horrible; the lexer produces surprising results for unterminated errors in all but this useless example; and the lexer uses a lot of buffer space and time to keep track of the contents of comments, which are then thrown away. This is the wrong engineering approach.

   Method 2 would say the above program has an unterminated comment. Method 2 is easier and more efficient in time and buffer space. It also recognizes errors more reasonably. This is a much more efficient and easier approach.

2. (15 points) These questions are about the handling of variables by a compiler for a simple language like C. In your answers, address only the compiler behavior that is *necessary for code generation.* Do not address type checking or other aspects of semantic analysis are not strictly necessary to emit code for correct programs.

   (a) Describe the compiler's processing of a global variable g of type `int`, both at the point of *declaration* and at the point of *use*.

   Solution:

   At the point of declaration, the compiler needs to enter info g into the symbol table, including its type, etc. but, especially, its location. The location is a constant offset relative to a section of memory reserved for global variables. The offset is established when the variable is *allocated* within the global data section. The actual address of

the global data section of memory is often not established until link time, but it is a constant when the program is actually executed.
At the point of use, the compiler needs to compute the address of the variable. If no array indexing is required (as when it is of type `int`, the address is a known constant, so no code needs to be generated. However, code is needed to fetch the value of the variable from the memory location.

(b) How is the handling of a local variable declaration of type `int` different from the handling of the global variable of the same type?

Solution:

Obviously, it is tagged as a local, not global variable when it goes in the symbol table. The offset is relative to a stack frame pointer of some kind, which is usually stored in a machine register. The stack frame pointer is set up when a function is called, and restored when the function returns.
When the variable is accessed, code is needed to add the contents of the frame pointer to the offset for the local variable. This is almost exactly the same code that would be required for an expression with + in it. Once the location of the variable has been computed, code must be generated to fetch the value.

(c) What information does the compiler have to maintain in the symbol table to generate code for `A[i].f[j]`?

Solution:

It needs to keep track of whether the variable `A` is local or global, and what its offset is (as above). It also needs remember the sizes of the array elements (to do array indexing) and the offsets of fields within structures. The generated code computes (frame pointer) + (offset of A) + i * (size of A elements) + (offset of f) + j * (size of A[i].f elements)

3. (10 points)

Why would it be useful for an optimizing compiler to have optimizations on an intermediate representation (such as 3-address code) *and* peephole optimization at the instruction level?

Solution:

The most sophisticated optimizations occur on intermediate code, because

- more information about the programmer's intent is available at the intermediate code level.
- they are more-or-less machine-independent, so they don't have to be rewritten when the compiler is retargeted.
- the intermediate representation can be designed to make the optimizations easy to implement.

Even with very sophisticated optimizations on intermediate code, peephole optimization is still useful. Some optimizations depend on information that is not available until the instructions are generated. For example, some machines have long jumps and short jumps, and won't know whether a short jump can be used until it knows the layout of the instructions.

4. (35 points) Consider the following context-free grammar:

$$\begin{array}{lcl} S & \rightarrow & Sa \\ S & \rightarrow & bS \\ S & \rightarrow & c \end{array}$$

(a) (3 points) Show that the grammar is ambiguous.

Solution:

There are two leftmost derivations for "bca": $S \overset{L}{\Longrightarrow} Sa \overset{L}{\Longrightarrow} bSa \overset{L}{\Longrightarrow} bca$
$S \overset{L}{\Longrightarrow} bS \overset{L}{\Longrightarrow} bSa \overset{L}{\Longrightarrow} bca$

(b) (10 points) Write the canonical collections of LR(1) items for this grammar.

Solution:

S'	$\rightarrow$	$\bullet S$, \$	S'	$\rightarrow$	$S\bullet$, \$
S	$\rightarrow$	$\bullet Sa$, \$a	S	$\rightarrow$	$S \bullet a$, \$a
S	$\rightarrow$	$\bullet bS$, \$a			
S	$\rightarrow$	$\bullet c$, \$a			

S	$\rightarrow$	$b \bullet S$, \$a	S	$\rightarrow$	$c\bullet$, \$a
S	$\rightarrow$	$\bullet Sa$, \$a			
S	$\rightarrow$	$\bullet bS$, \$a			
S	$\rightarrow$	$\bullet c$, \$a			

S	$\rightarrow$	$Sa\bullet$, \$a	S	$\rightarrow$	$bS\bullet$, \$a
			S	$\rightarrow$	$S \bullet a$, \$a

(c) (2 points) Identify all conflicting items, and the types of the conflicts (e.g., "shift-reduce conflict in state 3 on d").

Solution: There is a shift/reduce conflict on a in this state:

$$\begin{array}{lcl} S & \rightarrow & bS\bullet, \$a \\ S & \rightarrow & S \bullet a, \$a \end{array}$$

This stems from the ambiguity in the grammar. E.g., should bca be parsed as $(bc)a$ (reduce at this state) or $b(ca)$ (shift the a).

(d) (5 points) Could the original grammar be converted into an LALR(1) parser that parses all input correctly by resolving conflicts, in the way that YACC and similar parser generators allow? If so, how should they be resolved? In either case, please explain (briefly).

Solution:

Yes. Surprisingly, we can always shift in the situation of the previous problem or always reduce. In either case, the correct language will be accepted, although different trees will result from the two different ways of resolving the conflict.

(e) (5 points) Rewrite the grammar in an equivalent form that is suitable for LL parsing and minimizes the use of stack space.

Solution:

$$\begin{array}{lcl} S & \rightarrow & BcA \\ B & \rightarrow & bB|\epsilon \\ A & \rightarrow & aA|\epsilon \end{array}$$

(f) (5 points) Rewrite the grammar in an equivalent form that is directly suitable for LR parsing (i.e., does not result in conflicts) and minimizes the use of stack space.

Solution:

$$
\begin{aligned}
S &\rightarrow BcA \\
B &\rightarrow Bb|\epsilon \\
A &\rightarrow Aa|\epsilon
\end{aligned}
$$

(g) (5 points) In your modified LL(1) grammar, show the sequence of stack contents and inputs when parsing the input $bbcaa$.

Solution:

(top) stack	parse	action
S$	bbcaa$	expand $S \rightarrow BcA$
BcA$	bbcaa$	expand $B \rightarrow bB$
bBcA$	bbcaa$	match
BcA$	bcaa$	expand $B \rightarrow bB$
bBcA$	bcaa$	match
BcA$	caa$	expand $B \rightarrow \epsilon$
cA$	caa$	match
A$	aa$	expand $A \rightarrow aA$
aA$	aa$	match
A$	a$	expand $A \rightarrow aA$
aA$	a$	match
A$	$	expand $A \rightarrow \epsilon$
$	$	accept

Computer Architecture Comprehensive Exam

Exam Instructions

Answer each of the questions included in the exam. Write all of your answers directly on the examination paper, including any work that you wish to be considered for partial credit. The examination is open-book, and you may make use of the text, handouts, your own course notes, and a calculator. You may use a computer of any kind but no network.

On equations: Wherever possible, make sure to include the equation, the equation rewritten with the numerical values, and the final solution. Partial credit will be weighted appropriately for each component of the problem, and providing more information improves the likelihood that partial credit can be awarded.

On writing code: Unless otherwise stated, you are free to use any of the assembly instructions listed in the Appendix at the back of the book, including pseudoinstructions. You do not need to optimize your MIPS code unless specifically instructed to do so.

On time: You will have one hour to complete this exam. Budget your time and try to leave some time at the end to go over your work. The point weightings correspond roughly to the time each problem is expected to take.

THE STANFORD UNIVERSITY HONOR CODE

The Honor Code is an undertaking of the students, individually and collectively:

(1) that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;

(2) that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

I acknowledge and accept the Honor Code.

Magic Number _____

		Score	Grader
1. Short Answer	(15)	_____	_____
2. ISA	(15)	_____	_____
3. Pipelining	(15)	_____	_____
4. Cache	(15)	_____	_____
Total	(60)	_____	

Problem 1: Short Answer (15 points)
Please provide short, concise answers.

(a) [3 points] Can a direct mapped cache sometimes have a higher hit rate than a fully associative cache with an LRU replacement policy (on the same reference pattern and with the same cache size)? If so, give an example. If not, explain why not?

(b) [3 points] Give two ways virtual memory address translation is useful even if the total size of virtual memory (summed over all programs) is guaranteed to be smaller than physical memory.

(c) [3 points] How does a data cache take advantage of spatial locality?

(d) [6 points] What is the advantage of using a virtually-indexed physically-tagged L1 cache (as opposed to physically-indexed and physically-tagged)? What constraints does this design put on the size of the L1 cache

Problem 2: Instruction Set Architecture (15 points)

At TGIF, you hear a computer architecture graduate student propose that MIPS would have been better if only it had allowed arithmetic and logical instructions to have a register-memory addressing mode. This new mode would allow the replacement of sequences like:

```
lw   $1,  0($n)
add  $2,  $2,  $1
```

with:

```
add  $2,  0($n)
```

The student waves his hands (one holding bread and one holding cheese) saying this can be accomplished with only a 5% increase in the clock cycle and no increase in CPI.

You are fascinated by the possibilities of this breakthrough. However, before you drop out to create a startup, you decide to use SPECint2000 to evaluate the performance because it contains your favorite three applications (gcc, gzip, and of course perl). Since you love computer architecture so much, you have an EE282 book that has the instruction set mix for SPECint2000 with you (table at right).

Instruction	Average
load	26%
store	10%
add	19%
sub	3%
mul	0%
compare	5%
load imm	2%
cond branch	12%
cond move	1%
jump	1%
call	1%
return	1%
shift	2%
and	4%
or	9%
xor	3%
other logical	0%

(a) [8 points] What percentage of loads must be eliminated for the machine with the new instruction to have at least the same performance?

(b) [4 points] Give an example of a code sequence where the compiler could not perform this replacement even though it matches the general pattern?

(c) [3 points] Considering the usual 5 stage MIPS pipeline, why might this new instruction be problematic to implement with no change in CPI?

Problem 3: Pipelining (15 points)

Consider the following code:

```
Loop:lw   $1,  0($2)
     addi $1,  $1,  1
     sw   $1,  0($2)
     addi $2,  $2,  4
     sub  $4,  $3,  $2
     bne  $4,  $0, Loop
```

Assume that the initial value or R3 is R2 + 396

This code snippet will be executed on a MIPS pipelined processor with a 5-stage pipeline. Branches are resolved in the decode stage and *do not* have delay slots. All memory accesses take 1 clock cycle.

In the following three parts, you will be filling out pipeline diagrams for the above code sequence. Please use acronyms F, D, X, M and W for the 5 pipeline stages. For all cases of forwarding, use arrows to connect the source and destination stages. Simulate at most 7 instructions, making one pass through the loop and performing the first instruction a second time.

(a) [5 points] Fill in the pipeline diagram below for the execution of the above code sequence *without* any forwarding or bypassing hardware but assuming a register read and a write in the same clock cycle "forwards" through the register file.

	Cycle																
Instruction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

(b) [5 points] Fill in the pipeline diagram below for the execution of the above code sequence *with* traditional pipeline forwarding:

	Cycle																
Instruction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

(c) [5 points] <u>Aggressively</u> rearrange the order of the instructions (data dependencies have to be preserved) so that the number of instructions/cycles needed to execute the code snippet is minimized. Fill in the following table with the rearranged instruction sequence assuming traditional pipeline forwarding like part (b):

	Cycle																
Instruction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Problem 4: Cache Performance (15 points)

The following problem concerns basic cache lookups.

- The memory is byte addressable.
- Memory accesses are to **1-byte words** (not 4-byte words).
- Physical addresses are 13 bits wide.
- The cache is 4-way set associative, with a 4-byte block size and 32 total lines.

In the following tables, **all numbers are given in hexadecimal**. The *Index* column contains the set index for each set of 4 lines. The *Tag* columns contain the tag value for each line. The *V* column contains the valid bit for each line. The *Bytes 0–3* columns contain the data for each line, numbered left-to-right starting with byte 0 on the left.

The contents of the cache are as follows:

4-way Set Associative Cache																							
Index	Tag	V	Bytes 0–3				Tag	V	Bytes 0–3				Tag	V	Bytes 0–3				Tag	V	Bytes 0–3			
0	84	1	ED	32	0A	A2	9E	0	BF	80	1D	FC	10	0	EF	9	86	2A	E8	0	25	44	6F	1A
1	18	1	03	3E	CD	38	E4	0	16	7B	ED	5A	02	0	8E	4C	DF	18	E4	1	FB	B7	12	02
2	84	0	54	9E	1E	FA	84	1	DC	81	B2	14	48	0	B6	1F	7B	44	89	1	10	F5	B8	2E
3	92	0	2F	7E	3D	A8	9F	0	27	95	A4	74	57	1	07	11	FF	D8	93	1	C7	B7	AF	C2
4	84	1	32	21	1C	2C	FA	1	22	C2	DC	34	73	0	BA	DD	37	D8	28	1	E7	A2	39	BA
5	A7	1	A9	76	2B	EE	73	0	BC	91	D5	92	28	1	80	BA	9B	F6	6B	0	48	16	81	0A
6	8B	1	5D	4D	F7	DA	29	1	69	C2	8C	74	B5	1	A8	CE	7F	DA	BF	0	FA	93	EB	48
7	84	1	04	2A	32	6A	96	0	B1	86	56	0E	CC	0	96	30	47	F2	91	1	F8	1D	42	30

(a) [3 points] The box below shows the format of a physical address. Indicate (by labeling the diagram) the fields that would be used to determine the following:

- O The block offset within the cache line
- I The cache index
- T The cache tag

12	11	10	9	8	7	6	5	4	3	2	1	0

(b) [5 points] For the given physical address, indicate the cache entry accessed and the cache byte value returned in hex. Indicate whether a cache miss occurs. If there is a cache miss, enter "-" for "Cache Byte returned".

Physical address: 0x0D74
Physical address format (one bit per box)

12	11	10	9	8	7	6	5	4	3	2	1	0

Physical memory reference:

Parameter	Value
Cache Offset (CO)	0x
Cache Index (CI)	0x
Cache Tag (CT)	0x
Cache Hit? (Y/N)	
Cache Byte returned	0x

Physical address: 0x0AEE
Physical address format (one bit per box)

12	11	10	9	8	7	6	5	4	3	2	1	0

Physical memory reference:

Parameter	Value
Cache Offset (CO)	0x
Cache Index (CI)	0x
Cache Tag (CT)	0x
Cache Hit? (Y/N)	
Cache Byte returned	0x

(c) [4 points] For the given contents of the cache, list all of the hex physical memory addresses that will hit in Set 7. To save space, you should express contiguous addresses as a range. For example, you would write the four addresses 0x1314, 0x1315, 0x1316, 0x1317 as 0x1314-0x1317.

Answer: _____

The following templates are provided as scratch space:

12	11	10	9	8	7	6	5	4	3	2	1	0

12	11	10	9	8	7	6	5	4	3	2	1	0

12	11	10	9	8	7	6	5	4	3	2	1	0

(d) [3 points] For the given contents of the cache, what is the probability (expressed as a percentage) of a cache hit when the physical memory address ranges between 0x1080 - 0x109F. Assume that all addresses are equally likely to be referenced.

Probability = ___%

The following templates are provided as scratch space:

12	11	10	9	8	7	6	5	4	3	2	1	0

12	11	10	9	8	7	6	5	4	3	2	1	0

Computer Architecture Comprehensive Exam Solutions

Exam Instructions

Answer each of the questions included in the exam. Write all of your answers directly on the examination paper, including any work that you wish to be considered for partial credit. The examination is open-book, and you may make use of the text, handouts, your own course notes, and a calculator. You may use a computer of any kind but no network.

On equations: Wherever possible, make sure to include the equation, the equation rewritten with the numerical values, and the final solution. Partial credit will be weighted appropriately for each component of the problem, and providing more information improves the likelihood that partial credit can be awarded.

On writing code: Unless otherwise stated, you are free to use any of the assembly instructions listed in the Appendix at the back of the book, including pseudoinstructions. You do not need to optimize your MIPS code unless specifically instructed to do so.

On time: You will have one hour to complete this exam. Budget your time and try to leave some time at the end to go over your work. The point weightings correspond roughly to the time each problem is expected to take.

THE STANFORD UNIVERSITY HONOR CODE

The Honor Code is an undertaking of the students, individually and collectively:

(1) that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;

(2) that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

I acknowledge and accept the Honor Code.

Magic Number _____

		Score	Grader
1. Short Answer	(15)	_____	_____
2. ISA	(15)	_____	_____
3. Pipelining	(15)	_____	_____
4. Cache	(15)	_____	_____
Total	(60)	_____	

Problem 1: Short Answer (15 points)
Please provide short, concise answers.

(a) [3 points] Can a direct mapped cache sometimes have a higher hit rate than a fully associative cache with an LRU replacement policy (on the same reference pattern and with the same cache size)? If so, give an example. If not, explain why not?

Answer:

Imagine a 4 word cache with an access pattern of 0, 1, 2, 3, 4, 0, 1, 2, 3, 4. The directed mapped cache will have a 30% hit rate while the LRU fully associative cache will have a 0% hit rate.

(b) [3 points] Give two ways virtual memory address translation is useful even if the total size of virtual memory (summed over all programs) is guaranteed to be smaller than physical memory.

Answer:

Some examples are:

1. *isolation: protect processes fro each others' memory*
2. *relocation: allow any program to run anywhere in physical memory*

(c) [3 points] How does a data cache take advantage of spatial locality?

Answer:

When a word is loaded from main memory, adjacent words are loaded into the cache line. Spatial locality says that these adjacent bytes are likely to be used. A common example iterating through elements in an array.

(d) [6 points] What is the advantage of using a virtually-indexed physically-tagged L1 cache (as opposed to physically-indexed and physically-tagged)? What constraints does this design put on the size of the L1 cache

Answer:

[3 points] The cache index can be derived from the virtual address without translation. This allows the cache line to be looked up in parallel with the TLB access that will provide the physical cache tag. This helps keep the cache hit time low in the common case where of a TLB hit

[3 points] For this to work, the cache index cannot be affected by the virtual to physical address translation. This would happen if the any of the bits from the cache index came from the virtual page number of the virtual address instead of the page offset portion of the address..

Problem 2: Instruction Set Architecture (15 points)

At TGIF, you hear a computer architecture graduate student propose that MIPS would have been better if only it had allowed arithmetic and logical instructions to have a register-memory addressing mode. This new mode would allow the replacement of sequences like:

```
lw   $1,  0($n)
add  $2,  $2,  $1
```

with:

```
add  $2,  0($n)
```

The student waves his hands (one holding bread and one holding cheese) saying this can be accomplished with only a 5% increase in the clock cycle and no increase in CPI.

You are fascinated by the possibilities of this breakthrough. However, before you drop out to create a startup, you decide to use SPECint2000 to evaluate the performance because it contains your favorite three applications (gcc, gzip, and of course perl). Since you love computer architecture so much, you have an EE282 book that has the instruction set mix for SPECint2000 with you (table at right).

Instruction	Average
load	26%
store	10%
add	19%
sub	3%
mul	0%
compare	5%
load imm	2%
cond branch	12%
cond move	1%
jump	1%
call	1%
return	1%
shift	2%
and	4%
or	9%
xor	3%
other logical	0%

(a) [8 points] What percentage of loads must be eliminated for the machine with the new instruction to have at least the same performance?

Answer:

NOTE: load imm (load immediate) is not a kind of lw instruction

Because the new design has a clock cycle time equal to 1.05 times the original clock cycle time, the new design must execute fewer instruction to achieve the same execution time. For the original design, we have:

$$CPU\ Time_{old} = CPI_{old} * CC_{old} * IC_{old}$$

For the new design, the equation is:

$$CPU\ Time_{new} = CPI_{new} * CC_{new} * IC_{new}$$
$$= CPI_{old} * (1.05*CC_{old}) * (IC_{old} - R)$$

where the CPI of the new design is the same as the original (as stated in the question), the new clock cycle is 5% longer, and the new design executes R fewer instructions than the original design. To find out how many loads must be removed to match the performance of the original design set the above two equations equal and solve for R:

$$CPI_{old} * CC_{old} * IC_{old} = CPI_{old} * (1.05*CC_{old}) * (IC_{old} - R)$$
$$R = 0.0476\ IC_{old}$$

Thus, the instruction count must decrease by 4.76% overall to achieve the same performance, and this 4.76% is comprised entirely of loads. The data shows that 26% of the gcc instruction mix is loads, so (4.76%/26%) = 18.3% of the loads must be replaced by the new register-memory instruction format for the performance of the old and new designs to be the same. If more than 18.3% of the loads can be replaced, then performance of the new design is better.

(b) [4 points] Give an example of a code sequence where the compiler could not perform this replacement even though it matches the general pattern?

Answer:

One example is when the $1 variable is reused in another instruction below. Another is when the initial sequence uses the same register as the load destination and the add destination instead of distinct registers.

(c) [3 points] Considering the usual 5 stage MIPS pipeline, why might this new instruction be problematic to implement with no change in CPI?

Answer:

The result from the load will be available after the 4^{th} stage (M) however it needs to be an input to the 3^{rd} stage (X).

Problem 3: Pipelining (15 points)

Consider the following code:

```
Loop:lw   $1,  0($2)
     addi $1,  $1,  1
     sw   $1,  0($2)
     addi $2,  $2,  4
     sub  $4,  $3,  $2
     bne  $4,  $0, Loop
```

Assume that the initial value or R3 is R2 + 396

This code snippet will be executed on a MIPS pipelined processor with a 5-stage pipeline. Branches are resolved in the decode stage and *do not* have delay slots. All memory accesses take 1 clock cycle.

In the following three parts, you will be filling out pipeline diagrams for the above code sequence. Please use acronyms F, D, X, M and W for the 5 pipeline stages. For all cases of forwarding, use arrows to connect the source and destination stages. Simulate at most 7 instructions, making one pass through the loop and performing the first instruction a second time.

(a) [5 points] Fill in the pipeline diagram below for the execution of the above code sequence *without* any forwarding or bypassing hardware but assuming a register read and a write in the same clock cycle "forwards" through the register file.

Answer:

	Cycle																
Instruction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
lw $1,0($2)	*F*	*D*	*X*	*M*	*W*												
addi $1,$1,1		*F*	*D*	*D*	*D*	*X*	*M*	*W*									
sw $1,0($2)			*F*	*F*	*F*	*D*	*D*	*D*	*X*	*M*	*W*						
addi $2,$2,4						*F*	*D*	*D*	*D*	*X*	*M*	*W*					
sub $4,$3,$2							*F*	*D*	*D*	*D*	*D*	*D*	*X*	*M*	*W*		
bne $4,$0,…								*F*	*F*	*F*	*F*	*F*	*D*	*D*	*D*		
lw $1,0($2)																*F*	*D*

(b) [5 points] Fill in the pipeline diagram below for the execution of the above code sequence *with* traditional pipeline forwarding:

Answer:

	Cycle																
Instruction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
`lw $1,0($2)`	*F*	*D*	*X*	*M*	*W*												
`addi $1,$1,1`		*F*	*D*	*D*	*X*	*M*	*W*										
`sw $1,0($2)`			*F*	*F*	*D*	*X*	*M*	*W*									
`addi $2,$2,4`					*F*	*D*	*X*	*M*	*W*								
`sub $4,$3,$2`						*F*	*D*	*X*	*M*	*W*							
`bne $4,$0,…`							*F*	*D*	*D*								
`lw $1,0($2)`										*F*	*D*	*X*	*M*	*W*			

(c) [5 points] Aggressively rearrange the order of the instructions (data dependencies have to be preserved) so that the number of instructions/cycles needed to execute the code snippet is minimized. Fill in the following table with the rearranged instruction sequence assuming traditional pipeline forwarding like part (b):

Answer:

	Cycle																
Instruction	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
`lw $1,0($2)`	*F*	*D*	*X*	*M*	*W*												
`addi $2,$2,4`		*F*	*D*	*X*	*M*	*W*											
`addi $1,$1,1`			*F*	*D*	*X*	*M*	*W*										
`sw $1,-4($2)`				*F*	*D*	*X*	*M*	*W*									
`sub $4,$3,$2`					*F*	*D*	*X*	*M*	*W*								
`bne $4,$0,…`						*F*	*D*										
`lw $1,0($2)`								*F*	*D*	*X*	*M*	*W*					

There was no way to aggressively rearrange the order without changing the offset of the sw instruction from 0 to -4 as shown in the above solution. Full credit was also given for saying it was impossible to reorder without changing the instructions.

Problem 4: Cache Performance (15 points)

The following problem concerns basic cache lookups.

- The memory is byte addressable.
- Memory accesses are to **1-byte words** (not 4-byte words).
- Physical addresses are 13 bits wide.
- The cache is 4-way set associative, with a 4-byte block size and 32 total lines.

In the following tables, **all numbers are given in hexadecimal**. The *Index* column contains the set index for each set of 4 lines. The *Tag* columns contain the tag value for each line. The *V* column contains the valid bit for each line. The *Bytes 0–3* columns contain the data for each line, numbered left-to-right starting with byte 0 on the left.

The contents of the cache are as follows:

4-way Set Associative Cache																						
Index	Tag	V	Bytes 0–3				Tag	V	Bytes 0–3				Tag	V	Bytes 0–3				Tag	V	Bytes 0–3			
0	84	1	ED	32	0A	A2	9E	0	BF	80	1D	FC	10	0	EF	9	86	2A	E8	0	25	44	6F	1A
1	18	1	03	3E	CD	38	E4	0	16	7B	ED	5A	02	0	8E	4C	DF	18	E4	1	FB	B7	12	02
2	84	0	54	9E	1E	FA	84	1	DC	81	B2	14	48	0	B6	1F	7B	44	89	1	10	F5	B8	2E
3	92	0	2F	7E	3D	A8	9F	0	27	95	A4	74	57	1	07	11	FF	D8	93	1	C7	B7	AF	C2
4	84	1	32	21	1C	2C	FA	1	22	C2	DC	34	73	0	BA	DD	37	D8	28	1	E7	A2	39	BA
5	A7	1	A9	76	2B	EE	73	0	BC	91	D5	92	28	1	80	BA	9B	F6	6B	0	48	16	81	0A
6	8B	1	5D	4D	F7	DA	29	1	69	C2	8C	74	B5	1	A8	CE	7F	DA	BF	0	FA	93	EB	48
7	84	1	04	2A	32	6A	96	0	B1	86	56	0E	CC	0	96	30	47	F2	91	1	F8	1D	42	30

(a) [3 points] The box below shows the format of a physical address. Indicate (by labeling the diagram) the fields that would be used to determine the following:

O The block offset within the cache line
I The cache index
T The cache tag

12	11	10	9	8	7	6	5	4	3	2	1	0
T	*T*	*T*	*T*	*T*	*T*	*T*	*T*	*I*	*I*	*I*	*O*	*O*

(b) [5 points] For the given physical address, indicate the cache entry accessed and the cache byte value returned in hex. Indicate whether a cache miss occurs. If there is a cache miss, enter "-" for "Cache Byte returned".

Physical address: 0x0D74
Physical address format (one bit per box)

12	11	10	9	8	7	6	5	4	3	2	1	0
0	*1*	*1*	*0*	*1*	*0*	*1*	*1*	*1*	*0*	*1*	*0*	*0*

Physical memory reference:

Parameter	Value
Cache Offset (CO)	*0x00*
Cache Index (CI)	*0x05*
Cache Tag (CT)	*0x6B*
Cache Hit? (Y/N)	*N*
Cache Byte returned	*0x-*

Physical address: 0x0AEE
Physical address format (one bit per box)

12	11	10	9	8	7	6	5	4	3	2	1	0
0	*1*	*0*	*1*	*0*	*1*	*1*	*1*	*0*	*1*	*1*	*1*	*0*

Physical memory reference:

Parameter	Value
Cache Offset (CO)	*0x02*
Cache Index (CI)	*0x03*
Cache Tag (CT)	*0x57*
Cache Hit? (Y/N)	*Y*
Cache Byte returned	*0xFF*

(c) [4 points] For the given contents of the cache, list all of the hex physical memory addresses that will hit in Set 7. To save space, you should express contiguous addresses as a range. For example, you would write the four addresses 0x1314, 0x1315, 0x1316, 0x1317 as 0x1314-0x1317.

Answer: *0x109C – 0x109F, 0x123C – 0x123F*

The following templates are provided as scratch space:

12	11	10	9	8	7	6	5	4	3	2	1	0
1	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*1*	*1*	*1*	-	-

12	11	10	9	8	7	6	5	4	3	2	1	0
1	*0*	*0*	*1*	*0*	*0*	*0*	*1*	*1*	*1*	*1*	-	-

12	11	10	9	8	7	6	5	4	3	2	1	0

(d) [3 points] For the given contents of the cache, what is the probability (expressed as a percentage) of a cache hit when the physical memory address ranges between 0x1080 - 0x109F. Assume that all addresses are equally likely to be referenced.

Probability = *50*%

The following templates are provided as scratch space:

12	11	10	9	8	7	6	5	4	3	2	1	0
1	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*

12	11	10	9	8	7	6	5	4	3	2	1	0
1	*0*	*0*	*0*	*0*	*1*	*0*	*0*	*1*	*1*	*1*	*1*	*1*

Stanford University
Computer Science Department

Fall 2005 Comprehensive Exam in Databases

1. **Open Book and Notes - NO laptop. Write your solutions in the spaces provided on the exam.**
2. **The exam is timed for one hour.**
3. **Write your Magic Number in the space provided on this sheet and the following sheet; DO NOT WRITE YOUR NAME.**

The following is a statement of the Stanford University Honor Code:

A. The Honor Code is an undertaking of the students, individually and collectively:
   1. that they will not give or receive aid in examinations; that they will not give or receive un-permitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
   2. that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

B. The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

C. While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

By writing my Magic Number below, I certify that I acknowledge and accept the Honor Code.

Magic Number----------

Stanford University Computer Science Department
2005 Comprehensive Exam in Databases

- The exam is *open book and notes.*
- There are 7 problems on the exam, with a varying number of points for each problem and subproblem for a total of 60 points (i.e., one point per minute). It is suggested that you look through the entire exam before getting started, in order to plan your strategy.
- Please write your solutions in the spaces provided on the exam. Make sure your solutions are neat and clearly marked.
- *Simplicity and clarity of solutions will count.* You may get as few as 0 points for a problem if your solution is far more complicated than necessary, or if we cannot understand your solution.

MAGIC NUMBER: _____

Problem	1	2	3	4	5	6	7	TOTAL
Max. points	12	8	6	8	10	8	8	60
Points								

1. *Relational Algebra* **(12 points)**. In this question, we use $t[X]$ to represent the components of tuple t in attributes X. Let relation R have schema XY (i.e., the set of attributes $X \cup Y$) and let relation S have schema YZ. Assume X, Y, and Z are disjoint. The *strong join* of R by S is the set of tuples $t[X]$ such that t is a tuple of R, and for all tuples r in R (including t) such that $r[X] = t[X]$, there is some tuple s in S such that $r[Y] = s[Y]$. Write, as a sequence of assignments of relational-algebra expressions to variables, a program that computes the strong join of $R(A, B)$ by $S(B, C)$. **Note**: you should use only the basic relational-algebra operators: select, project, union, intersection, difference, product, natural join, and theta-join. Also, please explain what each step is doing, if you want to be considered for partial credit.

2. *Functional Dependences* **(8 points)**. Armstrong's axioms for functional dependencies are:

   (a) *Reflexivity*: If $X \subseteq Y$, then $Y \rightarrow X$.

   (b) *Augmentation*: If $X \rightarrow Y$, then $XZ \rightarrow YZ$.

   (c) *Transitivity*: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$.

   From these axioms, prove the *union rule*: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$. Note that X, Y, and Z are sets of attributes, XY denotes $X \cup Y$, and nothing about disjointness is assumed.

3. *ODL and E/R Models* **(6 points).** Your answers to (a) and (b) will be judged not only on being truthful, but also on selecting among all options the things that are most significant.

   (a) **(4 points)** Name two things that object-oriented models such as ODL offer the designer and that are not found in the E/R model.

   (b) **(2 points)** Name one thing that the E/R model offers that ODL does not.

4. *XML* **(8 points).** Here is a DTD:

```
<!DOCTYPE a [
   <!ELEMENT a (b+, c*)>
   <!ELEMENT b (c?)>
   <!ELEMENT c (#PCDATA)>
]>
```

   (a) **(3 points)** Give a shortest (fewest number of elements) example of a sequence of opening and closing tags that is valid for this DTD.

   (b) **(5 points)** Give a shortest (fewest number of elements) example of a well-formed sequence of opening and closing tags where `a` is the root tag, `b` and `c` tags are also present, but the sequence is *not* valid for this DTD.

5. *Multivalued Dependencies and SQL Constraints* **(10 points)**. Consider a table $R(A, B, C)$. Make no assumptions about keys.

   (a) **(6 points)** Using SQL Assertions, write an assertion stating that the multivalued dependency $A \twoheadrightarrow B$ holds on R.

   (b) **(4 points)** Can you enforce the same dependency using SQL Tuple-Based `CHECK` constraints? If so, show the constraint(s). If not, explain why not.

6. *SQL Equivalence* **(8 points).** Consider the following two SQL queries over tables $R(A, B)$ and $S(C)$. Make no assumptions about keys.

```
Q1: select A from R                    Q2: select distinct A from R, S
    where B in (select C from S)           where R.B = S.C
```

Are these two queries equivalent? That is, do they return the same answer on all possible instances of R and S? If so, justify why. If not, show a counterexample.

7. *Transactions* **(8 points).** Consider a relation `Emp(ID, salary)` storing employee IDs and salaries, where ID is a key. Consider the following two transactions:

```
T1: begin transaction
    update Emp set salary = 2*salary where ID = 25
    update Emp set salary = 3*salary where ID = 25
    commit

T2: begin transaction
    update Emp set salary = 100 where salary > 100
    commit
```

Suppose the salary of the employee with ID=25 is 100 before either transaction executes.

(a) **(4 points)** If both transactions T1 and T2 execute to completion with isolation level *serializable*, what are the possible final salaries for the employee with ID=25?

(b) **(4 points)** Now suppose transaction T1 executes with isolation level *read-committed*, transaction T2 executes with isolation level *read-uncommitted*, and both transactions execute to completion. What are the possible final salaries for the employee with ID=25?

Stanford University Computer Science Department
2005 Comprehensive Exam in Databases
SAMPLE SOLUTION

1. *Relational Algebra* **(12 points)**. In this question, we use $t[X]$ to represent the components of tuple t in attributes X. Let relation R have schema XY (i.e., the set of attributes $X \cup Y$) and let relation S have schema YZ. Assume X, Y, and Z are disjoint. The *strong join* of R by S is the set of tuples $t[X]$ such that t is a tuple of R, and for all tuples r in R (including t) such that $r[X] = t[X]$, there is some tuple s in S such that $r[Y] = s[Y]$. Write, as a sequence of assignments of relational-algebra expressions to variables, a program that computes the strong join of $R(A, B)$ by $S(B, C)$. **Note**: you should use only the basic relational-algebra operators: select, project, union, intersection, difference, product, natural join, and theta-join. Also, please explain what each step is doing, if you want to be considered for partial credit.

   $T_1 := R - \pi_{A,B}(R \bowtie S)$
   $T_2 := \pi_A(T_1) \bowtie R$
   Answer := $\pi_A(R) - \pi_A(T_2)$

2. *Functional Dependences* **(8 points)**. Armstrong's axioms for functional dependencies are:

   (a) *Reflexivity*: If $X \subseteq Y$, then $Y \rightarrow X$.

   (b) *Augmentation*: If $X \rightarrow Y$, then $XZ \rightarrow YZ$.

   (c) *Transitivity*: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$.

   From these axioms, prove the *union rule*: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$. Note that X, Y, and Z are sets of attributes, XY denotes $X \cup Y$, and nothing about disjointness is assumed.

   (1) From $X \rightarrow Y$ get $XZ \rightarrow YZ$ by Augmentation

   (2) From $X \rightarrow Z$ get $X \rightarrow XZ$ by Augmentation

   (3) From (1) and (2) get $X \rightarrow YZ$ by Transitivity

3. *ODL and E/R Models* **(6 points)**. Your answers to (a) and (b) will be judged not only on being truthful, but also on selecting among all options the things that are most significant.

   (a) **(4 points)** Name two things that object-oriented models such as ODL offer the designer and that are not found in the E/R model.

   - Methods
   - Complex data types

   (b) **(2 points)** Name one thing that the E/R model offers that ODL does not.

   - Multiway relationships

4. *XML* **(8 points)**. Here is a DTD:

```
<!DOCTYPE a [
   <!ELEMENT a (b+, c*)>
   <!ELEMENT b (c?)>
   <!ELEMENT c (#PCDATA)>
]>
```

   (a) **(3 points)** Give a shortest (fewest number of elements) example of a sequence of opening and closing tags that is valid for this DTD.

```
<a> <b> </b> </a>
```

   (b) **(5 points)** Give a shortest (fewest number of elements) example of a well-formed sequence of opening and closing tags where `a` is the root tag, `b` and `c` tags are also present, but the sequence is *not* valid for this DTD.

```
<a> <c> </c> <b> </b> </a>
```

5. *Multivalued Dependencies and SQL Constraints* **(10 points)**. Consider a table $R(A, B, C)$. Make no assumptions about keys.

   (a) **(6 points)** Using SQL Assertions, write an assertion stating that the multivalued dependency $A \twoheadrightarrow B$ holds on R.

```
create assertion MVD check (
  not exists (
    select * from R R1, R R2
    where R1.A = R2.A
    and (R1.A, R1.B, R2.C) not in (select * from R)))
```

(b) **(4 points)** Can you enforce the same dependency using SQL Tuple-Based `CHECK` constraints? If so, show the constraint(s). If not, explain why not.

Although the above general assertion can be translated directly to a tuple-based check constraint, it will only be checked on insertions and updates to R. Since deletions to R can cause an MVD violation, a tuple-based check cannot enforce an MVD.

6. *SQL Equivalence* **(8 points)**. Consider the following two SQL queries over tables $R(A, B)$ and $S(C)$. Make no assumptions about keys.

```
Q1: select A from R                Q2: select distinct A from R, S
    where B in (select C from S)       where R.B = S.C
```

Are these two queries equivalent? That is, do they return the same answer on all possible instances of R and S? If so, justify why. If not, show a counterexample.

Not equivalent
Let $R = \{\langle 1, 2\rangle, \langle 1, 2\rangle\}$ and $S = \{2\}$
Q1 returns $\{1, 1\}$ while Q2 returns $\{1\}$

7. *Transactions* **(8 points)**. Consider a relation `Emp(ID,salary)` storing employee IDs and salaries, where ID is a key. Consider the following two transactions:

```
T1: begin transaction
    update Emp set salary = 2*salary where ID = 25
    update Emp set salary = 3*salary where ID = 25
    commit

T2: begin transaction
    update Emp set salary = 100 where salary > 100
    commit
```

Suppose the salary of the employee with ID=25 is 100 before either transaction executes.

(a) **(4 points)** If both transactions T1 and T2 execute to completion with isolation level *serializable*, what are the possible final salaries for the employee with ID=25?

100, 600

(b) **(4 points)** Now suppose transaction T1 executes with isolation level *read-committed*, transaction T2 executes with isolation level *read-uncommitted*, and both transactions execute to completion. What are the possible final salaries for the employee with ID=25?

100, 300, 600

Computer Graphics Comprehensive Exam

Computer Science Department
Stanford University
Fall 2005

NAME:

Note: This is exam is *closed-book*.

The exam consists of 5 questions. Each question is worth 20 points. Please answer all the questions in the space provided, overflowing on to the back of the page if necessary.

You have 60 minutes to complete the exam.

1. [20 points] Computer graphics definitions.

Define in a few sentences each of the following computer graphics terms. Some of these terms may be used in other fields, so be sure to give the computer graphics meaning.

1A [5 points] Control points.

1B [5 points] Depth cueing.

1C [5 points] Tessellation.

1D [5 points] Dithering.

2. [20 points] Window to viewport transformation.

Computer graphics systems implement abstractions for windows and viewports on top of the physical framebuffer. A framebuffer is a 2D array of pixels. For example, an SXGA framebuffer has size 1024 by 768. The pixel in the upper left hand corner of the framebuffer is (0,0) and the pixel at the lower right hand corner is (1023,767). A viewport is a rectangular subset of the framebuffer. For example, the user may place the viewport so its upper left hand corner is at (128,128) and it has a size of (512, 512). A window is a logical coordinate system defined by the user. For example, the user will say the window has an x range of (-1.0..1.0) meaning that the left edge of the viewport has coordinate -1 .0 and the right edge is 1.0. Similarly for y.

2A [10 points]. Suppose you have a window with coordinates (wxmin, wxmax,wymin, wymax) and a viewport with coordinates (vxmin, vxmax, vymin, vxmax). Derive a transformation that converts a point (x,y) in window coordinates to a point (x',y') in viewport coordinates.

2B [10 points]. The framebuffer in turn is stored in addressable memory. Memory addresses are linear; that is, 1-dimensional. Suppose the SXGA framebuffer is stored beginning at memory address `base`. Assuming the above window and viewport transformation, what is the transformation from window coordinates (x,y) to memory addresses?

3. [20 points] Window systems.

We all use window systems with overlapping windows everyday. By overlapping we mean that there are many windows visible and they are stacked on top of each other. Front windows can partially cover windows behind them.

Windows are normally associated with processes. Processes are allowed to draw into the windows that they own, or that they have permission to draw into. The graphic system must prevent processes from drawing into windows unless they have permission to do so.

Suppose you are implementing a drawing engine for common primitives like triangles, lines, points, characters etc. Describe a technique for ensuring that the process can only draw into the visible parts of the windows that it has permission to draw into. State your assumptions about how windows are represented in the system. Present the reasons you think your solution is efficient.

4. [20 points] Color and Transparency.

In computer graphics RGB triplets are used to both model light itself and the interaction of light with materials. For example, a light source emits light with color (Re,Ge,Be). Light from different sources can be added together. For example, if we have two light sources, the light from the two sources may be combined additively (Re+Re',Ge+Ge',Be+Be').

When light interacts with a material, some of the light may be absorbed. As an approximation, we can model the material as a filter with transmission coefficients (Rf,Gf,Bf). The transmission coefficients must be in the range 0 to 1. If light with energy (Re,Ge,Be) interacts with a filter with transmission coefficients (Rf,Gf,Bf) the amount of transmitted light is (Re*Rf,Ge*Gf,Be*Bf).

4A [7 points]. Suppose you want to draw a picture consisting of a stack of *n* transmissive surfaces in front of a single emissive surface. Write a formula for the visible light given the amount of light emitted and the filter coefficients of the *n* surfaces.

4B [7 points]. Suppose each surface in the stack both emits light and transmits light. Write a formula for the amount of visible light.

4C [6 points]. Suppose these surfaces are contained "within a pixel." That is, we consider stack of surfaces where each surface is the portion of the surface intersected by the pixel. Now, since surfaces have finite size, only a fraction of the pixel will be covered by the surface. Call that fraction α. Write a formula for the amount of visible light coming from a stack of transmitting and reflecting surfaces where each surface only covers a fraction of the pixel.

5 [20 points] Point inside triangle.

A modern rasterization system accepts as input triangles and produces as output a set of pixel fragments. The output pixels are all those inside the triangle. In order to do this, it needs an algorithm to determine whether a point is inside a triangle.

5A [15 points]. Suppose the triangle is given by 3 vertices (x1,y1), (x2,y2), and (x3,y3). Describe an algorithm that tests whether a point (x,y) is inside a triangle. Justify your algorithm.

5B [5 points]. A tricky aspect of point-inside-triangle algorithms are the boundary cases. For example, a point may be exactly on an edge. This is bad if multiple adjacent triangles are drawn, since points on the shared edges are drawn twice. Briefly describe how you would modify your algorithm to prevent points on the boundaries from being drawn multiple times. Make sure that the point is at least drawn once!

Computer Graphics Comprehensive Exam

Computer Science Department
Stanford University
Fall 2005

NAME:

Note: This is exam is *closed-book*.

The exam consists of 5 questions. Each question is worth 20 points. Please answer all the questions in the space provided, overflowing on to the back of the page if necessary.

You have 60 minutes to complete the exam.

1. [20 points] Computer graphics definitions.

Define in a few sentences each of the following computer graphics terms. Some of these terms may be used in other fields, so be sure to give the computer graphics meaning.

1A [5 points] Control points.

Continuous functions, curves and surfaces may be formed from a discrete set of points. These points are called the control points. Control points are used to for Bezier and B-Spline curves for example.

1B [5 points] Depth cueing.

Depth cueing is a technique for modulating the intensity or saturation of a color as a function of depth.

Depth cues are any technique used to provide information about depth. This is a more general usage.

1C [5 points] Tessellation.

Tesselation is a technique for converting a smooth surface into a set of polygons, quadrilaterals or triangles.

Tesselation also refers to tiling the plane with polygons. This is a more general usage.

1D [5 points] Dithering.

Dithering is a technique for approximating continuous greys by making a pattern of black and white dots. Dithering can be generalized to work with color, to to use dots of different values.

2. [20 points] Window to viewport transformation.

Computer graphics systems implement abstractions for windows and viewports on top of the physical framebuffer. A framebuffer is a 2D array of pixels. For example, an SXGA framebuffer has size 1024 by 768. The pixel in the upper left hand corner of the framebuffer is (0,0) and the pixel at the lower right hand corner is (1023,767). A viewport is a rectangular subset of the framebuffer. For example, the user may place the viewport so its upper left hand corner is at (128,128) and it has a size of (512, 512). A window is a logical coordinate system defined by the user. For example, the user will say the window has an x range of (-1.0..1.0) meaning that the left edge of the viewport has coordinate -1 .0 and the right edge is 1.0. Similarly for y.

2A [10 points]. Suppose you have a window with coordinates (wxmin, wxmax,wymin, wymax) and a viewport with coordinates (vxmin, vxmax, vymin, vxmax). Derive a transformation that converts a point (x,y) in window coordinates to a point (x',y') in viewport coordinates.

x' = (x – wxmin) * (vxmax – vxmin)/(wxmax – wxmin) + vxmin

y' = (y – wymin) * (vymax – vymin)/(wymax – wymin) + vymin

2B [10 points]. The framebuffer in turn is stored in addressable memory. Memory addresses are linear; that is, 1-dimensional. Suppose the SXGA framebuffer is stored beginning at memory address `base`. Assuming the above window and viewport transformation, what is the transformation from window coordinates (x,y) to memory addresses?

addr = base + (1024 * y + x) * bytesperpixel

3. [20 points] Window systems.

We all use window systems with overlapping windows everyday. By overlapping we mean that there are many windows visible and they are stacked on top of each other. Front windows can partially cover windows behind them.

Windows are normally associated with processes. Processes are allowed to draw into the windows that they own, or that they have permission to draw into. The graphic system must prevent processes from drawing into windows unless they have permission to do so.

Suppose you are implementing a drawing engine for common primitives like triangles, lines, points, characters etc. Describe a technique for ensuring that the process can only draw into the visible parts of the windows that it has permission to draw into. State your assumptions about how windows are represented in the system. Present the reasons you think your solution is efficient.

This is an open-ended questions. Common correct answers included:

1) Create an off-screen buffer for each window and composite them in back to front order.

2) Assign a window-id to each pixel and only write to a pixel if the window-id assigned to the drawing process is equal to the window-id of that pixel.

3) Maintain a clipping polygon that describes the visible part of each window and clip each primitive to that polygon.

In all approaches, you needed to describe how you prevented drawing in regions outside your window.

4. [20 points] Color and Transparency.

In computer graphics RGB triplets are used to both model light itself and the interaction of light with materials. For example, a light source emits light with color (Re,Ge,Be). Light from different sources can be added together. For example, if we have two light sources, the light from the two sources may be combined additively (Re+Re',Ge+Ge',Be+Be').

When light interacts with a material, some of the light may be absorbed. As an approximation, we can model the material as a filter with transmission coefficients (Rf,Gf,Bf). The transmission coefficients must be in the range 0 to 1. If light with energy (Re,Ge,Be) interacts with a filter with transmission coefficients (Rf,Gf,Bf) the amount of transmitted light is (Re*Rf,Ge*Gf,Be*Bf).

4A [7 points]. Suppose you want to draw a picture consisting of a stack of *n* transmissive surfaces in front of a single emissive surface. Write a formula for the visible light given the amount of light emitted and the filter coefficients of the *n* surfaces.

R = Re (Prod Rf_i) where Rf_ is the red transmission coefficient for the ith surface.

4B [7 points]. Suppose each surface in the stack both emits light and transmits light. Write a formula for the amount of visible light.

R_i = Re_i (Prod_j<i Rf_j)

Attenuate the light from surface i by all the filters in front of it.

R = sum_i R_i

4C [6 points]. Suppose these surfaces are contained "within a pixel." That is, we consider stack of surfaces where each surface is the portion of the surface intersected by the pixel. Now, since surfaces have finite size, only a fraction of the pixel will be covered by the surface. Call that fraction α. Write a formula for the amount of visible light coming from a stack of transmitting and reflecting surfaces where each surface only covers a fraction of the pixel.

Update emitted light for each surface

Re_i' = α_i Re_i
Rf_i' = α_i Rf_i + (1-α_i)

And then use the formula in 4B

5 [20 points] Point inside triangle.

A modern rasterization system accepts as input triangles and produces as output a set of pixel fragments. The output pixels are all those inside the triangle. In order to do this, it needs an algorithm to determine whether a point is inside a triangle.

5A [15 points]. Suppose the triangle is given by 3 vertices (x1,y1), (x2,y2), and (x3,y3). Describe an algorithm that tests whether a point (x,y) is inside a triangle. Justify your algorithm.

1. Form the line equation for each edge of the triangle

$A\,x + B\,y + C < 0$ if the point x, y contains the interior of the triangle

$A = (y1 - y2)$
$B = (x2 - x1)$
$C = (x2*y1-x1*y2)$

These equations must be formed consistently by taking pairs of points as you move counterclockwise or clockwise around the triangle.

2. A point is inside the triangle if it is inside all three lines. A point is inside the half-space formed by a line if the line equation evaluates to a negative quantity.

5B [5 points]. A tricky aspect of point-inside-triangle algorithms are the boundary cases. For example, a point may be exactly on an edge. This is bad if multiple adjacent triangles are drawn, since points on the shared edges are drawn twice. Briefly describe how you would modify your algorithm to prevent points on the boundaries from being drawn multiple times. Make sure that the point is at least drawn once!

A tie-breaker must be introduced for points on the line. There are various ways to do this.

One of the most common is to test whether the half-space lines to the left or the right of the line. If the half-space lies to the left, points on the line are considered inside; if the half-space lies to the right, they are outside. The side of the half-space can be found by looking at the normal to the line, or the direction that the line moves (for example, $x2 > x1$). The tie-breaking rule needs to work for horizontal lines as well. For example, if the line is horizontal, then the upper half-space is inside, and the lower half-space is outside.

Comprehensive Examination in Logic
November 2005

30 questions

Time: 1 hour

Instructions

- **Do not open the test booklet until instructed to do so.**
- The exam is open book and open notes, but no laptops or electronic accessories are allowed.
- Answer each question in the booklet itself. The answers should fit the space given. Writing on the margin/footer will **not** be considered.
- It is strongly recommended that you work out the answer outside the test booklet before answering.
- All questions have penalties for wrong answers. Read the instructions carefully before you start.
- **THE HONOR CODE:**

  1. The honor code is an undertaking of the students individually and collectively:
     (a) that they will not give or receive aid in examinations; they will not give or receive unauthorized aid in class work, in the preparation of reports, or in any other work that is to be used by the instructors as the basis of grading;
     (b) that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the honor code.
  2. The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will avoid, as far as possible, academic procedures that create temptation to violate the Honor code.
  3. While the faculty alone have the right and the obligation to set academic requirements, the students and the faculty will work together to establish optimal conditions for honorable academic work.

  By writing my "magic–number" below, I acknowledge and accept the honor code.

WRITE MAGIC NUMBER: _____

Some conventions: We assume a few pieces of notation: a first-order logic sentence has no free variables, while a first-order formula may have some free variables (i.e., sentences are formulas with no free variables).

Given a set of first-order sentences Σ, we denote by $\mathcal{M}(\Sigma)$ the class of models of all the sentences in Σ. Given a set of structures (also called interpretations) M, we use $Th(M)$ to denote the set of sentences satisfied by all structures in M. Given a set of sentences Σ the set of consequences of Σ, $Th(\mathcal{M}(\Sigma))$ is denoted by $Cn(\Sigma)$.

A theory is a set of sentences. A theory $\mathcal{T}$ is called axiomatizable if for some recursively enumerable set of sentences (called axioms) $Cn(\Sigma) = \mathcal{T}$.

Grading: For both sections A and B:

correct answer	2 points each
no answer	0 points each
first 5 incorrect answers (in A and B combined)	0 points each
second 5 incorrect answers (in A and B combined)	-2 points
incorrect answers beyond 10 (in A and B combined)	-4 points

Section A
For each question in this section you need to choose one out the four possible choices provided. Indicate your answer by writing your choice clearly in the box provided. Remember: **No points deducted for leaving questions unanswered**.

1. Which of the following is a propositional tautology?

   (a) $((P \to Q) \to P) \to P$.

   (b) $P \to (P \to (P \to Q))$.

   (c) $(P \to (Q \vee R)) \leftrightarrow \neg((Q \wedge R) \to P)$.

   (d) None of the above.

2. Which of the following is true about complete sets of propositional connectives?

   (a) The set $\{\to, \neg\}$ is **not** a complete set of connectives.

   (b) There is some binary propositional connective (expressed for example as a truth table) that is complete by itself.

   (c) the constant **false** or **true** must always be present in a complete set of connectives.

   (d) The set $\{\wedge, \vee\}$ is a complete set of connectives.

3. Let $\mathcal{F}, \mathcal{G}$ stand for arbitrary sentences of propositional logic. What is the relationship between these three statements?

   I $\mathcal{F}$ is equivalent to $\mathcal{G}$.

   II $(\mathcal{F} \equiv \mathcal{G})$ is valid.

   III $\mathcal{F}$ is valid precisely when $\mathcal{G}$ is valid.

   (a) (I), (II) and (III) are equivalent.

   (b) (I), (II) and (III) are not all equivalent, but (I) implies (II), (II) implies (III).

   (c) (I), (II) and (III) are not all equivalent, but (II) implies (III), (III) implies (I).

   (d) none of the above.

4. Given the sentence $\sigma \ : \ \exists x \, [p(x) \ \wedge \neg \, p(x)]$, which of the following is correct?

   (a) σ is valid in first-order logic.

   (b) σ is valid in first-order logic, but no tableau proof can be found.

   (c) σ is not valid in any theory.

   (d) σ is not valid in first-order logic, but it is valid in some axiomatic theory.

5. Recall that a well-founded relation R over a domain D is a binary relation $R \subseteq D \times D$, such that no infinite chain $a_1, a_2, \ldots$ of elements exists with $(a_i, a_{i+1}) \in R$.

   Let R be a well-founded relation and let R^{-1} be its inverse relation (i.e., $R^{-1} = \{(a, b) \mid (b, a) \in R\}$.) Which of the following is true?

   (a) R^{-1} is always a well-founded relation.

   (b) if the domain is finite, then R^{-1} is a well-founded relation.

   (c) if the domain is not finite, then R^{-1} cannot be a well-founded relation.

   (d) R^{-1} is a well-founded relation iff R is a finite set.

6. Let $\mathcal{T}$ be a theory that is axiomatizable and complete. Which of the following is necessarily true?

   (a) $\mathcal{T}$ is decidable.

   (b) $\mathcal{T}$ is decidable only if it has a finite axiomatization.

   (c) $\mathcal{T}$ is decidable only if it is consistent.

   (d) $\mathcal{T}$ is decidable only if it is inconsistent.

7. Let $\mathcal{N}$ be the structure of the natural numbers with 0, addition, successor, multiplication, and less-than relation. Which of the following is true?

   (a) The theory $Th(\mathcal{N})$ has a finite model.

   (b) The theory $Th(\mathcal{N})$ has a model with uncountable cardinality.

   (c) The theory $Th(\mathcal{N})$ has a finite axiomatization.

   (d) all of the above.

8. Let t and s be two terms, and s_1 be a proper subterm of s (i.e., s_1 is a subterm of s different from s). Which of the following is true?

   (a) s can be unified with s_1.

   (b) if t can be unified with s, then t cannot be unified with s_1.

   (c) if t can be unified with s_1, then t cannot be unified with s.

   (d) t may be unifiable with both s and s_1.

9. Let P be an arbitrary unary predicate. Consider a formula φ and the following first-order logic sentence σ:

$$\exists * \left[(\neg P(x) \vee \varphi) \rightarrow (\neg\varphi \wedge P(x))\right]$$

   Note that for each choice of formula φ, a different sentence σ is generated.

   Which of the following is necessarily true?

   (a) for some φ, the sentence σ is valid.

   (b) for some φ, the sentence σ is satisfiable, while for some others σ is unsatisfiable.

   (c) for all φ, the sentence σ is satisfiable.

   (d) for all φ, the sentence σ is unsatisfiable.

Section B

For each question in this section you need to circle "T" if you think the statement holds or "F" if you think it does not. Remember: **There is no penalty for leaving questions unanswered.**

1. Consider an *arbitrary* propositional logic sentence $\mathcal{F}$. If $\mathcal{F}$ is not valid, then we can prove that $(\neg\ \mathcal{F})$ is valid using the deductive tableau method.

T	F

2. Consider arbitrary closed sentences $\mathcal{F}$ and $\mathcal{G}$ in first-order logic, and a set Σ of axioms. If $\mathcal{F}$ is valid in the theory if Σ (i.e., $\mathcal{F}$ is in $Cn(\Sigma)$), and $(\neg\mathcal{F})$ is valid in the theory of Σ, then $\mathcal{G}$ is valid in the theory of Σ.

T	F

3. If two first-order sentences are valid then they are equivalent.

T	F

4. If two sentences are equivalent in a theory $\mathcal{T}$, then they are also equivalent in the theory consisting of all the first-order sentences not in $\mathcal{T}$.

T	F

5. If the resolution rule is applied with a unifier θ that is not a most general unifier (m.g.u.) then soundness is compromised (i.e., some non-valid formula can be proved valid using a deductive tableau).

T	F

6. If a theory $\mathcal{T}$ is finitely axiomatizable, then there is some sentence σ such that, if $\varphi \in \mathcal{T}$, then $\sigma \vDash \varphi$.

T	F

7. A (not necessarily finite) set Σ of sentences has a model iff **every** finite subset $\Sigma_0 \subseteq \Sigma$ has a model.

T	F

8. Let Γ be an arbitrary set of first-order sentences, and ψ be a first-order sentence. Then either $\Gamma \models \psi$ **or** $\Gamma \models \neg\psi$ (or both).

T	F

9. Let ψ be an arbitrary first-order sentence. There is some set of sentences Γ for which $\Gamma \models \psi$ **and** $\Gamma \models \neg\psi$.

T	F

10. The finite union of well-founded relations is a well-founded relation.

T	F

11. The intersection of well-founded relations is a well-founded relation.

T	F

12. The composition of well-founded relations is a well-founded relation.

T	F

13. Consider the first-order language consisting of equality and a binary predicate symbol E. You can think of it as the language of directed graphs.

    Does the following predicate R defines reachability?

    (reachability means that for every interpretation $\mathcal{I}$ and two elements in its domain $a, b \in |\mathcal{I}|$ if $R^{\mathcal{I}}(a, b)$ then there are some $a_1, \ldots, a_n$ with $E^{\mathcal{I}}(a_i, a_{i+1})$ and $a_1 = a$, $a_n = b$.)

$$\forall x, y\, [R(x, y) \equiv \Big(E(x, y) \vee \exists z\, (R(x, z) \wedge E(z, y))\Big)]$$

T	F

14. If a sentence is valid then it must always occur with positive polarity in any enclosing sentence.

T	F

15. Let $\mathcal{F}$ be a sentence that is not valid, which contains a subsentence $\mathcal{G}$ with at least one occurrence with negative polarity. Replacing **all** occurrences of $\mathcal{G}$ in $\mathcal{F}$ by **true** can not generate a valid sentence.

T	F

16. The set of non-valid sentences of first-order logic is **not** recursively enumerable.

T	F

17. Let $\mathcal{T}$ be a decidable theory. If $\mathcal{T}$ is consistent, then it is complete.

T	F

18. Let $\mathcal{T}$ be a decidable and consistent theory and Σ an axiomatization of $\mathcal{T}$. There is a way to extend Σ to a complete and still consistent theory by adding one of ψ or $\neg\psi$ for all sentences for which $\Sigma \nvdash \psi$ and $\Sigma \nvdash \neg\psi$.

T	F

19. Let Γ be a theory. If for all models $\mathcal{I}$ of Γ either $\vDash_{\mathcal{I}} \varphi$ or $\vDash_{\mathcal{I}} \neg\varphi$, then Γ is complete.

T	F

20. Let φ be a sentence over a first-order language, and $\mathcal{T}_1$ and $\mathcal{T}_2$ be two theories over the same language. If $\mathcal{T}_1 \vDash \varphi$ and $\mathcal{T}_2 \vDash \varphi$ then certainly $\mathcal{T}_1 \cup \mathcal{T}_2 \vDash \varphi$.

T	F

21. Let φ be a sentence over a first-order language and $\mathcal{T}_1$ and $\mathcal{T}_2$ be two theories over the same language. If $\mathcal{T}_1 \nvdash \varphi$ and $\mathcal{T}_2 \nvdash \varphi$ then certainly $\mathcal{T}_1 \cup \mathcal{T}_2 \nvdash \varphi$.

T	F

22. If a theory $\mathcal{T}$ is finitely axiomatizable, then for every positive number k there is an axiomatization of $\mathcal{T}$ with exactly k sentences.

T	F

23. Let $\mathcal{F}$ be a sentence with quantifiers. The following holds *if and only if* $\mathcal{F}$ is valid: *Validity-preserving skolemization transforms $\mathcal{F}$ into a sentence without quantifiers $\mathcal{G}$ that is equivalent to $\mathcal{F}$.*

T	F

24. The validity preserving skolemization of

$$\exists x\,\forall z\,\exists y\,\big(p(x,y)\,\wedge\neg p(y,f(z))\big)$$

is **equivalent** to

$$\big(p(x',y')\,\wedge\neg p(y',f(f(x')))\big).$$

T	F

25. Is the following a first-order validity?

$$\big([\forall x\,p(x)\rightarrow\exists x\,(Q(x)\equiv\neg Q(f(f(x))))]\rightarrow\forall y\,p(y)\big)\rightarrow\forall z\,p(z)$$

T	F

26. If a propositional deductive tableau rule is sound, then the generated sub-tableau must be valid.

T	F

27. If $\mathcal{F}[P]$ is unsatisfiable then $\mathcal{F}[\neg P]$ is satisfiable.

T	F

28. Let a and b be constants, and x, y and z be variables. The tuple

$$\langle f(z,g(z,a)),f(x,g(b,y)),f(b,g(b,x))\rangle$$

is **not** unifiable.

T	F

29. If $\forall*\,(\mathcal{F}\vee\mathcal{G})$ is valid, then either $\exists*\,\mathcal{F}$ is valid or $\exists*\,\mathcal{G}$ is valid (or both).

T	F

30. Let $\mathcal{F}$ be valid if and only if $\mathcal{G}$ is valid, and let $\mathcal{H}'$ be obtained from $\mathcal{H}$ by replacing all occurrences of $\mathcal{F}$ by $\mathcal{G}$ in $\mathcal{H}$. Then $\mathcal{H}$ is valid **iff** $\mathcal{H}'$ is.

T	F

SOLUTIONS Comp. Exam. in Logic
November 2005

39 questions

Time: 1 hour

Instructions

- **Do not open the test booklet until instructed to do so.**
- The exam is open book and open notes, but no laptops or electronic accessories are allowed.
- Answer each question in the booklet itself. The answers should fit the space given. Writing on the margin/footer will **not** be considered.
- It is strongly recommended that you work out the answer outside the test booklet before answering.
- All questions have penalties for wrong answers. Read the instructions carefully before you start.
- **THE HONOR CODE:**

  1. The honor code is an undertaking of the students individually and collectively:
     (a) that they will not give or receive aid in examinations; they will not give or receive unauthorized aid in class work, in the preparation of reports, or in any other work that is to be used by the instructors as the basis of grading;
     (b) that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the honor code.
  2. The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will avoid, as far as possible, academic procedures that create temptation to violate the Honor code.
  3. While the faculty alone have the right and the obligation to set academic requirements, the students and the faculty will work together to establish optimal conditions for honorable academic work.

  By writing my "magic–number" below, I acknowledge and accept the honor code.

WRITE MAGIC NUMBER: _____

Some conventions: We assume a few pieces of notation: a first-order logic sentence has no free variables, while a first-order formula may have some free variables (i.e., sentences are formulas with no free variables).

Given a set of first-order sentences Σ, we denote by $\mathcal{M}(\Sigma)$ the class of models of all the sentences in Σ. Given a set of structures (also called interpretations) M, we use $Th(M)$ to denote the set of sentences satisfied by all structures in M. Given a set of sentences Σ the set of consequences of Σ, $Th(\mathcal{M}(\Sigma))$ is denoted by $Cn(\Sigma)$.

A theory is a set of sentences. A theory $\mathcal{T}$ is called axiomatizable if for some recursively enumerable set of sentences (called axioms) $Cn(\Sigma) = \mathcal{T}$.

Grading: For both sections A and B:

correct answer	2 points each
no answer	0 points each
first 5 incorrect answers (in A and B combined)	0 points each
second 5 incorrect answers (in A and B combined)	-2 points
incorrect answers beyond 10 (in A and B combined)	-4 points

Section A
For each question in this section you need to choose one out the four possible choices provided. Indicate your answer by writing your choice clearly in the box provided. Remember: **No points deducted for leaving questions unanswered**.

1. Which of the following is a propositional tautology?

   (a) $((P \to Q) \to P) \to P$.

   (b) $P \to (P \to (P \to Q))$.

   (c) $(P \to (Q \vee R)) \leftrightarrow \neg((Q \wedge R) \to P)$.

   (d) None of the above.

   a

2. Which of the following is true about complete sets of propositional connectives?

   (a) The set $\{\to, \neg\}$ is **not** a complete set of connectives.

   (b) There is some binary propositional connective (expressed for example as a truth table) that is complete by itself.

   (c) the constant **false** or **true** must always be present in a complete set of connectives.

   (d) The set $\{\wedge, \vee\}$ is a complete set of connectives.

   b

3. Let $\mathcal{F}, \mathcal{G}$ stand for arbitrary sentences of propositional logic. What is the relationship between these three statements?

   I $\mathcal{F}$ is equivalent to $\mathcal{G}$.

   II $(\mathcal{F} \equiv \mathcal{G})$ is valid.

   III $\mathcal{F}$ is valid precisely when $\mathcal{G}$ is valid.

   (a) (I), (II) and (III) are equivalent.

   (b) (I), (II) and (III) are not all equivalent, but (I) implies (II), (II) implies (III).

   (c) (I), (II) and (III) are not all equivalent, but (II) implies (III), (III) implies (I).

   (d) none of the above.

   b

4. Given the sentence $\sigma : \exists x\,[p(x) \wedge \neg\, p(x)]$, which of the following is correct?

   (a) σ is valid in first-order logic.

   (b) σ is valid in first-order logic, but no tableau proof can be found.

   (c) σ is not valid in any theory.

   (d) σ is not valid in first-order logic, but it is valid in some axiomatic theory.

   d

5. Recall that a well-founded relation R over a domain D is a binary relation $R \subseteq D \times D$, such that no infinite chain $a_1, a_2, \ldots$ of elements exists with $(a_i, a_{i+1}) \in R$.

   Let R be a well-founded relation and let R^{-1} be its inverse relation (i.e., $R^{-1} = \{(a, b) \mid (b, a) \in R\}$.) Which of the following is true?

   (a) R^{-1} is always a well-founded relation.

   (b) if the domain is finite, then R^{-1} is a well-founded relation.

   (c) if the domain is not finite, then R^{-1} cannot be a well-founded relation.

   (d) R^{-1} is a well-founded relation iff R is a finite set.

   b

6. Let $\mathcal{T}$ be a theory that is axiomatizable and complete. Which of the following is necessarily true?

   (a) $\mathcal{T}$ is decidable.

   (b) $\mathcal{T}$ is decidable only if it has a finite axiomatization.

   (c) $\mathcal{T}$ is decidable only if it is consistent.

   (d) $\mathcal{T}$ is decidable only if it is inconsistent.

   a

7. Let $\mathcal{N}$ be the structure of the natural numbers with 0, addition, successor, multiplication, and less-than relation. Which of the following is true?

   (a) The theory $Th(\mathcal{N})$ has a finite model.

   (b) The theory $Th(\mathcal{N})$ has a model with uncountable cardinality.

   (c) The theory $Th(\mathcal{N})$ has a finite axiomatization.

   (d) all of the above.

b

8. Let t and s be two terms, and s_1 be a proper subterm of s (i.e., s_1 is a subterm of s different from s). Which of the following is true?

   (a) s can be unified with s_1.

   (b) if t can be unified with s, then t cannot be unified with s_1.

   (c) if t can be unified with s_1, then t cannot be unified with s.

   (d) t may be unifiable with both s and s_1.

d

9. Let P be an arbitrary unary predicate. Consider a formula φ and the following first-order logic sentence σ:

$$\exists * \left[(\neg P(x) \vee \varphi) \rightarrow (\neg\varphi \wedge P(x))\right]$$

   Note that for each choice of formula φ, a different sentence σ is generated.

   Which of the following is necessarily true?

   (a) for some φ, the sentence σ is valid.

   (b) for some φ, the sentence σ is satisfiable, while for some others σ is unsatisfiable.

   (c) for all φ, the sentence σ is satisfiable.

   (d) for all φ, the sentence σ is unsatisfiable.

b

Section B

For each question in this section you need to circle "T" if you think the statement holds or "F" if you think it does not. Remember: **There is no penalty for leaving questions unanswered.**

1. Consider an *arbitrary* propositional logic sentence $\mathcal{F}$. If $\mathcal{F}$ is not valid, then we can prove that $(\neg\, \mathcal{F})$ is valid using the deductive tableau method.

	F

2. Consider arbitrary closed sentences $\mathcal{F}$ and $\mathcal{G}$ in first-order logic, and a set Σ of axioms. If $\mathcal{F}$ is valid in the theory of Σ (i.e., $\mathcal{F}$ is in $Cn(\Sigma)$), and $(\neg\mathcal{F})$ is valid in the theory of Σ, then $\mathcal{G}$ is valid in the theory of Σ.

T	

3. If two first-order sentences are valid then they are equivalent.

T	

4. If two sentences are equivalent in a theory $\mathcal{T}$, then they are also equivalent in the theory consisting of the Cn of all the first-order sentences not in $\mathcal{T}$.

	F

5. If the resolution rule is applied with a unifier θ that is not a most general unifier (m.g.u.) then soundness is compromised (i.e., some non-valid formula can be proved valid using a deductive tableau).

	F

6. If a theory $\mathcal{T}$ is finitely axiomatizable, then there is some sentence σ such that, if $\varphi \in \mathcal{T}$, then $\sigma \vDash \varphi$.

T	

7. A (not necessarily finite) set Σ of sentences has a model iff **every** finite subset $\Sigma_0 \subseteq \Sigma$ has a model.

T	

8. Let Γ be an arbitrary set of first-order sentences, and ψ be a first-order sentence. Then either $\Gamma \vDash \psi$ **or** $\Gamma \vDash \neg\psi$ (or both).

	F

9. Let ψ be an arbitrary first-order sentence. There is some set of sentences Γ for which $\Gamma \vDash \psi$ **and** $\Gamma \vDash \neg\psi$.

T	

10. The finite union of well-founded relations is a well-founded relation.

	F

11. The intersection of well-founded relations is a well-founded relation.

T	

12. The composition of well-founded relations is a well-founded relation.

	F

13. Consider the first-order language consisting of equality and a binary predicate symbol E. You can think of it as the language of directed graphs.

    The following predicate R defines reachability.

    (reachability means that for every interpretation $\mathcal{I}$ and two elements in its domain $a, b \in |\mathcal{I}|$ if $R^{\mathcal{I}}(a, b)$ then there are some $a_1, \ldots, a_n$ with $E^{\mathcal{I}}(a_i, a_{i+1})$ and $a_1 = a$, $a_n = b$.)

$$\forall x, y\,[R(x, y) \equiv \Big(E(x, y) \vee \exists z\,(R(x, z) \wedge E(z, y))\Big)]$$

	F

14. If a sentence is valid then it must always occur with positive polarity in any enclosing sentence.

	F

15. Let $\mathcal{F}$ be a sentence that is not valid, which contains a subsentence $\mathcal{G}$ with at least one occurrence with negative polarity. Replacing **all** occurrences of $\mathcal{G}$ in $\mathcal{F}$ by **true** can not generate a valid sentence.

	F

16. The set of non-valid sentences of first-order logic is **not** recursively enumerable.

T	

17. Let $\mathcal{T}$ be a decidable theory. If $\mathcal{T}$ is consistent, then it is complete.

	F

18. Let $\mathcal{T}$ be a decidable and consistent theory and Σ an axiomatization of $\mathcal{T}$. There is a way to extend Σ to a complete and still consistent theory by adding one of ψ or $\neg\psi$ for all sentences for which $\Sigma \nvDash \psi$ and $\Sigma \nvDash \neg\psi$.

T	

19. Let Γ be a theory. If for all models $\mathcal{I}$ of Γ either $\vDash_{\mathcal{I}} \varphi$ or $\vDash_{\mathcal{I}} \neg\varphi$, then Γ is complete.

	F

20. Let φ be a sentence over a first-order language, and $\mathcal{T}_1$ and $\mathcal{T}_2$ be two theories over the same language. If $\mathcal{T}_1 \vDash \varphi$ and $\mathcal{T}_2 \vDash \varphi$ then certainly $\mathcal{T}_1 \cup \mathcal{T}_2 \vDash \varphi$.

T	

21. Let φ be a sentence over a first-order language and $\mathcal{T}_1$ and $\mathcal{T}_2$ be two theories over the same language. If $\mathcal{T}_1 \nvDash \varphi$ and $\mathcal{T}_2 \nvDash \varphi$ then certainly $\mathcal{T}_1 \cup \mathcal{T}_2 \nvDash \varphi$.

	F

22. If a theory $\mathcal{T}$ is finitely axiomatizable, then for every positive number k there is an axiomatization of $\mathcal{T}$ with exactly k sentences.

T	

23. Let $\mathcal{F}$ be a sentence with quantifiers. The following holds *if and only if* $\mathcal{F}$ is valid: *Validity-preserving skolemization transforms $\mathcal{F}$ into a sentence without quantifiers $\mathcal{G}$ that is equivalent to $\mathcal{F}$.*

	F

24. The validity preserving skolemization of

$$\exists x\,\forall z\,\exists y\,\big(p(x,y)\,\wedge\neg p(y,f(z))\big)$$

is **equivalent** to

$$\big(p(x',y')\,\wedge\neg p(y',f(f(x')))\big).$$

	F

25. Is the following a first-order validity?

$$\big([\forall x\,p(x)\rightarrow\exists x\,(Q(x)\equiv\neg Q(f(f(x))))]\rightarrow\forall y\,p(y)\big)\rightarrow\forall z\,p(z)$$

T	

26. If a propositional deductive tableau rule is sound, then the generated sub-tableau must be valid.

	F

27. If $\mathcal{F}[P]$ is unsatisfiable then $\mathcal{F}[\neg P]$ is satisfiable.

	F

28. Let a and b be constants, and x, y and z be variables. The tuple

$$\langle f(z,g(z,a)),f(x,g(b,y)),f(b,g(b,x))\rangle$$

is **not** unifiable.

T	

29. If $\forall*\,(\mathcal{F}\vee\mathcal{G})$ is valid, then either $\exists*\,\mathcal{F}$ is valid or $\exists*\,\mathcal{G}$ is valid (or both).

	F

30. Let $\mathcal{F}$ be valid if and only if $\mathcal{G}$ is valid, and let $\mathcal{H}'$ be obtained from $\mathcal{H}$ by replacing all occurrences of $\mathcal{F}$ by $\mathcal{G}$ in $\mathcal{H}$. Then $\mathcal{H}$ is valid **iff** $\mathcal{H}'$ is.

	F

Stanford University
Computer Science Department

Fall 2005 Comprehensive Exam in Networks

1. **Closed Book, - NO laptop. Write only in the Blue Book provided.**
2. **The exam is timed for one hour.**
3. **Write your Magic Number on this sheet and the Blue Book; DO NOT WRITE YOUR NAME.**

The following is a statement of the Stanford University Honor Code:

A. The Honor Code is an undertaking of the students, individually and collectively:
   1. that they will not give or receive aid in examinations; that they will not give or receive un-permitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
   2. that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

B. The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

C. While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

By writing my Magic Number below, I certify that I acknowledge and accept the Honor Code.

Magic Number----------

Comprehensive Exam: Networks (60 points)
Closed Book: Autumn 2005

1. (15 points total) *End to end*

   (a) (5 points) Define the so-called "end-to-end principle" as applied to the Internet.

   (b) (5 points) In the deep slumber of the night, the ghost of Albert Einstein comes to you, exhorting you to not take the end-to-end principle on faith, but to realize that it can be justified based on probabilistic analysis. That is, a given probability of undetected failure can be achieved at a lower cost using an end-to-end design rather than the alternative(s?). Give an example of a simple end-to-end design and probalistic analysis that supports Albert's position, or else argue that this is nonsense (perhaps induced by Albert's discomfort with entanglement).

   (c) (5 points) There is a known major risk to high winds coming up when fighting forest fires in steep terrain. In the summer of 1994 in Glenwood Springs, Colo., 13 firefighters died tragically when the wind came up in a steep canyon in which they were fighting a forest fire (mirroring a similar trajedy in 1949). In the book "Fire on the Mountain," John N. Maclean makes the case that bureaucratic bungling led to a revised weather report at the weather bureau not making it to these firefighters to warn them to leave the area. How would an "end-to-end" firefighter operate, and how might that affect Maclean's analysis?

2. (15 points total) *Transport Protocol Design*

   (a) (8 points) TCP maintains two key dynamic values per connection: 1) a round-trip estimation and 2) congestion window size. For each, describe in brief: i) how it is computed, ii) how is it used, iii) how it can be confused, and iv) what is done to minimize this confusion. (Your answers to iii) and iv) can be included in i) as long as you make clear what the challenges are in computing each accurately and the techniques used to address them, and you can focus on any particular published algorithms for i), if you are aware of several.)

(b) (7 points) Imagine that Osama Bin Laden (OBL) has gained control of the backbone Internet routers, but the Mullahs have ordered (for some curious reason) that he is limited to just being able to drop up to K percent of the packets, but can decide which ones to drop, at least based on L3/L4 properties. (The fatwa says he is not allowed to drop more or inject or modify traffic, thank you, Mullahs!) Describe your analysis of what OBL's best strategies are for interfering with Internet usage and the minimum values of K required to do so.

3. (15 points total) *Network Routing*

   (a) (7 points) Describe an example network that includes alternative routes between several different nodes with different costs, and show how distance-vector routing works on this network. Illustrate how distance-vector routing can behave badly on this network, giving a specific failure scenario.

   (b) (8 Points) Describe how BGP avoids loops as well as the tradeoff it makes between stability, scalability, lowest cost routes and policy, illustrating with an example.

4. (15 points total) *Ethernet*

   (a) (6 points) The phrase "Ethernet spanning tree loop" strikes terror into the hearts of LAN network operators everywhere. Describe what this is, why it arises, and whether or not such "loops" occur at the IP level, justifying your answer.

   (b) (5 points) Draw a graph of the throughput of a local area network as a function of (increasing) offered load, indicating its behavior depending on whether it is using pure Aloha, slotted Aloha or CSMA-CD as its access protocol. Justify the difference in performance characteristics for each.

   (c) (4 points) Peterson and Davie say: "It might seem that a wireless protocol would follow exactly the same CSMA-CD algorithm as Ethernet" as a lead-in to why not. Describe why not and what 802.11 does about it.

The End

Computer Science Department
Stanford University
Comprehensive Examination in Numerical Analysis
Fall 2005

1. **Solution of polynomial equations [20pts]**

Let $p_n(x)$ be a polynomial of degree n with real coefficients. Assume the roots are distinct.

(a) [5pts] Define Newton's Method for finding the roots of $p_n(x)$.

(b) [7pts] Show the rate of convergence for finding the roots.

(c) [8pts] Suppose $p_n(x) = (x-a)^k q_{n-k}(x)$, where $q_{n-k}(x)$ is a polynomial of degree $n-k$ and assume all its roots are distinct. Show how to modify Newton's Method so that the root a is obtained and convergence is quadratic.

2. **Solution of linear equations [20pts]**

(a) [7pts] Describe Gaussian elimination as a matrix factorization. What can be said about the magnitude of the elements of the factorization when partial pivoting is implemented?

(b) [8pts] Assume we have two methods for implementing Gaussian elimination so that we have

$$A = L_1 U_1 \text{ and } A = L_2 U_2.$$

Show that $L_1 = L_2$ and $U_1 = U_2$.

(c) [5pts] Suppose, we know

$$A = LU \text{ and } B = AD$$

where D is a diagonal matrix with non-zero diagonal elements. What is the relation between the LU factorization of A and that of B?

3. **Differential equations [20pts]**

Consider the differential equation

$$(*) \quad y' = f(x, y)$$
$$y(a) = \alpha.$$

There are two methods that are frequently used for developing methods for solving differential equations. They are as follows:

i.) Formulas based on quadrature.

ii.) Methods based on Taylor Series.

(a) [13pts] Develop Euler's Method from each of these two methods for solving the differential equation given by $(*)$.

(b) [7pts] Discuss the error: $|y_n - y(x_n)|$ where y_n is the numerical solution to the difference equation and $y(x_n)$ is the exact solution at x_n. Show how it depends on the mesh width: h.

Comprehensive Exam: Programming Languages Autumn 2005

This is a 60-minute closed-book exam and the point total for all questions is 60.

All of the intended answers may be written within the space provided. (*Do not use a separate blue book.*) Succinct answers that do not include irrelevant observations are preferred. You may use the back of the preceding page for scratch work. If you to use the back side of a page to write part of your answer, be sure to mark your answer clearly.

The following is a statement of the Stanford University Honor Code:

A. *The Honor Code is an undertaking of the students, individually and collectively:*

   (1) *that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;*

   (2) *that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.*

B. *The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.*

C. *While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.*

By writing my "magic number" below, I certify that I acknowledge and accept the Honor Code.

_____

(Number)

Prob	# 1	# 2	# 3	# 4	Total
Score					
Max	20	12	12	16	60

10/31/05 REVISION

1. (*20 points*) ..... Short Answer

Answer each question in a few words or phrases.

(a) (***4 points***) In C++, it is possible to designate that a member function `f` is `virtual` in a base class, but not write this designation in the derived class. What relationship between a C++ base class `vtable` and derived class `vtable` implies that the function `f` is also virtual in the derived class? Explain.

(b) (***4 points***) Explain why this program, if compiled and executed without type checking, produces a run-time type error. At which point (and which line of code) will the Java compiler or run-time system report the error?

```
1: class A { ... amethod ...}
2:      class B extends A { ... bmethod ...}
3:      B[ ] bArray = new B[10]
4:      A[ ] aArray = bArray
5:      aArray[0] = new A()
6:      bArray[0].bmethod()
```

(c) (*4 points*) List three different reasons why Java programs generally run slower than C++ programs.

(d) (*4 points*) What is the main advantage of tail recursion optimization? Is it time or space saving? Explain.

(e) (*4 points*) The designers of C+++ are considering allowing functions to be declared inside any block. As a consequence, a function could be declared inside one function, and passed or returned to another function. What implementation costs are associated with this design option?

2. (*12 points*) ..... Scope and parameter passing and activation records

Consider this simple program:

```
1   int a :=  1
2   int b := 10
3   proc f(m) {
4      a := a+m
5      b := a+m
6   }
7   proc g(n) {
8      int a := 5
9      f(n)
10  }
11  g(a)
12  print a , b
```

(a) (*6 points*) Assume that all parameters are passed using call-by-value.

i. What values are printed on line 12 under static scoping?

ii. What values are printed under dynamic scoping?

(b) (*6 points*) Assume that only static scoping is used.

i. What values are printed if all arguments are passed using call-by-reference?

ii. What if call-by-value-result is used instead?

3. (*12 points*) ..... Templates and Generics

(a) (*4 points*) Consider a C++ class of this form:

```
template <class T>
class C<T> {
   ...
int f (T *x, T *y){ ...  x->less(y) ...}
   ...
};
```

Suppose a C++ program contains a an object of type `C<A>` for some C++ class `A`. Explain what functions, operators, and so on, class `A` must define in order for the code shown to compile and link correctly.

(b) (*4 points*) Suppose that an analogous class `C` is written in Java.

```
class C<T> {
   ...
int f (T x, T y){ ...  x.compareTo(y) ...}
   ...
}
```

Explain why the Java 1.5 compiler would not accept the code, based on the fragments shown.

(c) (*4 points*) The Java code can be written so that it will compile-time type check. Instead of `class C<T>`, we will need to write something of the form

`class C<T extends ...>`

What would you write in the underlined region to make this correct Java? Explain, and describe any additional classes or interfaces you may need.

```
class C<T extends  ____________________________> {
   ...
int f (T x, T y){ ...  x.compareTo(y) ...}
   ...
}
```

4. (*16 points*) ..... Concurrency

For the following question, we will use a very simple linked list class designed to be used by multiple threads. The class stores elements in a linked list and whenever an element is added or deleted, the code finds the maximum element in the list and stores it in a member variable. Assume that `max()` will never be called on an empty list.

```
public class CLinkedList
{
  protected CElement head_; //The list's head
  Comparable max_; //A reference to the maximum list element

  public CLinkedList()
  { //Constructor
    head_ = null;
    max_ = null;
  }

  //Assume max() will never be called on an empty list
  public synchronized Comparable max()
  {
    return max_;
  }

  //Add an element to the front of the list
  public synchronized void addElement(Comparable element)
  {
    CElement newHead = new CElement(element, head_);
    head_ = newHead;
    updateMax(); //Choose a new max element, if necessary
  }

  //Remove the first element from the list.
  public synchronized Comparable deleteFirst()
  {
    CElement deletedElement = head_;
    head_ = deletedElement.next; //Remove the first element.
    deletedElement.next = null; //Removed element points nowhere
    updateMax();                //Update the max element
    return deletedElement.element;
  }

  //Scans each element in the list until it finds an equal element or
  //the end of the list.  Returns true if the element is found.
  public boolean lookup(Comparable o){
    for (CElement i = head_; i != null; i = i.next){
      if (o.equals(i.element)){
        return true;
      }
    }
    return false;
  }

  //Scan every element in the list to find the maximum element.  Update
  //the max_ member to reference the maximum element.
  private void updateMax()
  {
```

```
  if (head_ != null){
    CElement tempMax = head_;
    for (CElement i = head_; i != null; i = i.next){
      if (tempMax.element.compareTo(i.element) < 0 ||
          tempMax.element.compareTo(i.element) == 0){
        tempMax = i;
      }
    }
    max_ = tempMax.element;
  }
}
//A simple list cell class.
private class CElement
{
  public CElement(Comparable element, CElement next){
    this.element = element;
    this.next = next;
  }
  public Comparable element; //The element referenced
                             //by this list element.
  public CElement next; //The next element in the linked list.
}
```

(a) (*5 points*) A race condition exists in the CLinkedList implementation because `lookup()` is not synchronized. In some cases, a call to `lookup()` may not find an object that is actually in the list. Give a sequence of calls to CLinkedList member functions by different threads that could result in `lookup()` failing to find an object that is in the list. Your answer should include a time-line of method calls made by specific threads and a description of what causes `lookup()` to fail.

(b) (*4 points*) Can a multi-threaded program become deadlocked when multiple threads concurrently operate on a single CLinkedList object? Explain.

(c) (*3 points*) If the programmer knows that CLinkedList will only be used on uniprocessor machines, is it necessary for `addElement()` and `deleteFirst()` to be synchronized?

(d) (***4 points***) Calling a synchronized method is more expensive than calling an unsynchronized method because a thread must acquire a lock before running in the monitor. If the programmer knows that `max()` will be called frequently, she would like to optimize the function by making it unsynchronized.

For this problem, keep in mind that the result returned by `max()` may be stale by the time the calling thread uses `max()`'s return value, which is correct. For example, thread 1 could call `max()` and get `max()`'s result, but before using the result thread 2 adds a new element to the list that becomes the new maximum value. The same line of reasoning applies to an unsynchronized version of `max()`.

i. Let's assume that the programmer knows that only `int` elements will be stored in CLinkedLists. Assuming that the programmer can rewrite any necessary code related to the type of elements stored in the linked list, can she make the `max()` method unsynchronized? Explain. For the purposes of this problem, assume that writes to integer variables are atomic, but writes to doubles are not atomic. If an operation is atomic, it either updates any necessary state and completes or it terminates without updating any state at all. If an operation is not atomic, it is possible that it can be interrupted in the middle of its execution and that some intermediate state might be visible to other parts of the program until it is resumed.

ii. If CLinkedList objects will hold only `double` values, can the programmer rewrite code and make `max()` unsynchronized? Explain.

Comprehensive Exam: Programming Languages Autumn 2005

1. (*20 points*) ..... Short Answer

   Answer each question in a few words or phrases.

   (a) (*4 points*) In C++, it is possible to designate that a member function `f` is `virtual` in a base class, but not write this designation in the derived class. What relationship between a C++ base class `vtable` and derived class `vtable` implies that the function `f` is also virtual in the derived class? Explain.

   Answer: In C++, it is essential that the `vtable` of a derived class match the `vtable` of the base class. Otherwise, a base class virtual function might not appear in a derived class `vtable`, and C++ subtyping would break.

   (b) (*4 points*) Explain why this program, if compiled and executed without type checking, produces a run-time type error. At which point (and which line of code) will the Java compiler or run-time system report the error?

```
1: class A { ... amethod ...}
2:     class B extends A { ... bmethod ...}
3:     B[ ] bArray = new B[10]
4:     A[ ] aArray = bArray
5:     aArray[0] = new A()
6:     bArray[0].bmethod()
```

   Answer: The program compiles due to Java array covariance. If executed without run-time checking, an error would occur on line 6 because the program attempts to call a class B method on a class A object. In normal Java execution, a run-time exception will be thrown by the assignment `aArray[0] = new A()` in line 5.

   (c) (*4 points*) List three different reasons why Java programs generally run slower than C++ programs.

   Answer: The overhead of interpreting bytecode instructions, array bounds checks, and run-time tests associated with type casts.

   (d) (*4 points*) What is the main advantage of tail recursion optimization? Is it time or space saving? Explain.

   Answer: Space savings. There is some time saving, associated with not having to allocated and deallocate activation records. However, the main consequence of tail recursion optimization is that the amount of space used by a function does not grow in proportion to the number of recursive calls.

   (e) (*4 points*) The designers of C+++ are considering allowing functions to be declared inside any block. As a consequence, a function could be declared inside one function, and passed or returned to another function. What implementation costs are associated with this design option?

   Answer: Functions must be represented by closures, and activation records can no longer be allocated/deallocated in a stack-like (last allocated/first deallocated) manner.

2. (*12 points*) ..... Scope and parameter passing and activation records

Consider this simple program:

```
1   int a :=  1
2   int b := 10
3   proc f(m) {
4      a := a+m
5      b := a+m
6   }
7   proc g(n) {
8      int a := 5
9      f(n)
10  }
11  g(a)
12  print a , b
```

(a) (*6 points*) Assume that all parameters are passed using call-by-value.

i. What values are printed on line 12 under static scoping?
Answer: 2, 3

ii. What values are printed under dynamic scoping?
Answer: 1, 7

(b) (*6 points*) Assume that only static scoping is used.

i. What values are printed if all arguments are passed using call-by-reference?
Answer: 2, 4

ii. What if call-by-value-result is used instead?
Answer: 1, 3

3. (*12 points*) ..... Templates and Generics

(a) (*4 points*) Consider a C++ class of this form:

```
template <class T>
class C<T> {
   ...
int f (T *x, T *y){ ...  x->less(y) ...}
   ...
};
```

Suppose a C++ program contains a an object of type `C<A>` for some C++ class `A`. Explain what functions, operators, and so on, class `A` must define in order for the code shown to compile and link correctly.

Answer: Class `A` must define a member function `less` in such a way that if `A *x` and `A *y` as declared in the argument list of `f`, then `x->less(y)` will type check and compile.

(b) (*4 points*) Suppose that an analogous class `C` is written in Java.

```
class C<T> {
   ...
int f (T x, T y){ ...  x.compareTo(y) ...}
   ...
}
```

Explain why the Java 1.5 compiler would not accept the code, based on the fragments shown.

Answer: This code will not compile because the compiler cannot determine that if `T x` as declared in the code, then `x` has method `compareTo`.

(c) (*4 points*) The Java code can be written so that it will compile-time type check. Instead of `class C<T>`, we will need to write something of the form

```
class C<T extends ...>
```

What would you write in the underlined region to make this correct Java? Explain, and describe any additional classes or interfaces you may need.

```
class C<T extends  ____________________________> {
   ...
int f (T x, T y){ ...  x.compareTo(y) ...}
   ...
}
```

Answer: A partial answer is that we need `T extends Comparable`, where `Comparable` is some class with a `compareTo` operation. However, a full answer should mention the type if `compareTo`. Although this could be a method that is applicable to any object, that doesn't give much of a way to implement interesting comparisons. It's better to define `Comparable` as a parameterized interface `Comparable(T)` with `T` the type of the argument to `compareTo`. Then use `T extends Comparable(T)` in the declaration of `C`.

4. (*16 points*) ..... Concurrency

For the following question, we will use a very simple linked list class designed to be used by multiple threads. The class stores elements in a linked list and whenever an element is added or deleted, the code finds the maximum element in the list and stores it in a member variable. Assume that `max()` will never be called on an empty list.

```
public class CLinkedList
{
  protected CElement head_; //The list's head
  Comparable max_; //A reference to the maximum list element

  public CLinkedList()
  { //Constructor
    head_ = null;
    max_ = null;
  }

  //Assume max() will never be called on an empty list
  public synchronized Comparable max()
  {
    return max_;
  }

  //Add an element to the front of the list
  public synchronized void addElement(Comparable element)
  {
    CElement newHead = new CElement(element, head_);
```

```
  head_ = newHead;
  updateMax(); //Choose a new max element, if necessary
}

//Remove the first element from the list.
public synchronized Comparable deleteFirst()
{
  CElement deletedElement = head_;
  head_ = deletedElement.next; //Remove the first element.
  deletedElement.next = null; //Removed element points nowhere
  updateMax();                //Update the max element
  return deletedElement.element;
}

//Scans each element in the list until it finds an equal element or
//the end of the list.  Returns true if the element is found.
public boolean lookup(Comparable o){
  for (CElement i = head_; i != null; i = i.next){
    if (o.equals(i.element)){
      return true;
    }
  }
  return false;
}

//Scan every element in the list to find the maximum element.  Update
//the max_ member to reference the maximum element.
private void updateMax()
{
  if (head_ != null){
    CElement tempMax = head_;
    for (CElement i = head_; i != null; i = i.next){
      if (tempMax.element.compareTo(i.element) < 0 ||
          tempMax.element.compareTo(i.element) == 0){
        tempMax = i;
      }
    }
    max_ = tempMax.element;
  }
}
//A simple list cell class.
private class CElement
{
  public CElement(Comparable element, CElement next){
    this.element = element;
    this.next = next;
  }
  public Comparable element; //The element referenced
                             //by this list element.
  public CElement next; //The next element in the linked list.
}
```

(a) (*5 points*) A race condition exists in the CLinkedList implementation because `lookup()` is not synchronized. In some cases, a call to `lookup()` may not find an object that is actually in the list. Give a sequence of calls to CLinkedList member functions by different threads that could result in `lookup()` failing to find an object that is in the list. Your answer should include a time-line of method calls made by

specific threads and a description of what causes `lookup()` to fail.

Answer: Yes, we can execute `lookup()`, it finds the first element, and then a new thread is scheduled which deletes the first element and sets the next pointer to null. Therefore, `lookup()` will not search the entire list. Essentially, by not synchronizing `lookup()` it can see the list in an inconsistent state.

Some people may be tempted to say that there is a race condition involving `updateMax()`, but this is not the case. `updateMax()` is a private member function, so we know that it will only be called by `addElement()` and `deleteFirst()`. When called by either of those two functions, the executing thread *retains* its lock on the CLinkedList object because it has not left the scope of the monitor. Therefore, it is not possible for one thread to be executing in `updateMax()` while another is executing and another thread simultaneously executing in either `addElement()` or `deleteFirst()`.

(b) (*4 points*) Can a multi-threaded program become deadlocked when multiple threads concurrently operate on a single CLinkedList object? Explain.

Answer: No, there is only one lock per CLinkedList object – the implicit lock that is associated with every Object in Java. Because of this, no thread can hold a lock while waiting for another lock, and therefore deadlock is not possible.

(c) (*3 points*) If the programmer knows that CLinkedList will only be used on uniprocessor machines, is it necessary for `addElement()` and `deleteFirst()` to be synchronized?

Answer: Yes, because threads can be preempted on uniprocessor machines.

(d) (*4 points*) Calling a synchronized method is more expensive than calling an unsynchronized method because a thread must acquire a lock before running in the monitor. If the programmer knows that `max()` will be called frequently, she would like to optimize the function by making it unsynchronized.

For this problem, keep in mind that the result returned by `max()` may be stale by the time the calling thread uses `max()`'s return value, which is correct. For example, thread 1 could call `max()` and get `max()`'s result, but before using the result thread 2 adds a new element to the list that becomes the new maximum value. The same line of reasoning applies to an unsynchronized version of `max()`.

i. Let's assume that the programmer knows that only `int` elements will be stored in CLinkedLists. Assuming that the programmer can rewrite any necessary code related to the type of elements stored in the linked list, can she make the `max()` method unsynchronized? Explain. For the purposes of this problem, assume that writes to integer variables are atomic, but writes to doubles are not atomic. If an operation is atomic, it either updates any necessary state and completes or it terminates without updating any state at all. If an operation is not atomic, it is possible that it can be interrupted in the middle of its execution and that some intermediate state might be visible to other parts of the program until it is resumed.

Answer: Yes, if max_ is an int, it will be updated atomically, so no matter when max_'s value is read, it will be consistent with some view of the linked list.

ii. If CLinkedList objects will hold only `double` values, can the programmer rewrite code and make `max()` unsynchronized? Explain.

Answer: No, `max()` can not be unsynchronized. Consider a case with two threads on a dual-CPU machine, thread 1 and thread 2. Thread 1 calls `max()` while thread 2 is simultaneously updating max_'s value. Thread 2 writes the first word of max_, thread 1 reads max_, and thread 2 finishes writing the last word of max_. Thread 1 ends up reading the value of max_ in an inconsistent

state, which gives an incorrect result. This situation can occur because writes to doubles are not atomic.

Stanford University
Computer Science Department

Fall 2005 Comprehensive Exam in Software Systems

1. **Open Book, - NO laptop. Write only in the Blue Book provided.**
2. **The exam is timed for one hour.**
3. **Write your Magic Number on this sheet and the Blue Book; DO NOT WRITE YOUR NAME.**

The following is a statement of the Stanford University Honor Code:

A. The Honor Code is an undertaking of the students, individually and collectively:

1. that they will not give or receive aid in examinations; that they will not give or receive un-permitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;

2. that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

B. The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

C. While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

By writing my Magic Number below, I certify that I acknowledge and accept the Honor Code.

Magic Number----------

Comprehensive Exam — Systems software

Fall 2005

November 9, 2005

Answer each of the following questions, and give a sentence or two justification for your answer (4 points each).

1. Write the shortest, non-reentrant legal C function you can and why you think it is non-reentrant.

2. Your machine can only do 32-bit loads and stores. What is the locking-related problem in the following code? How would you fix it?

```
struct foo {
        lock_t a_lock; // always held before touching a
        lock_t b_lock; // always held before touching b

        char a;
        char b;
};
...

void update_ab(struct foo *f) {
        lock(f->a_lock);
        f->a++;
        unlock(f->a_lock);

        lock(f->b_lock);
        f->b++;
        unlock(f->b_lock);
}
```

3. Your threaded code has no race conditions. It does have this routine:

```
foo()
        lock(a);
        lock(b);
        ...
        unlock(a);
        ...
```

```
lock(a);
...
unlock(b);
unlock(b);
}
```

Ignore performance: what is wrong here?

4. In what way is a good proportional-share process scheduling algorithm essentially equivalent to a good graphics line-drawing algorithm? (Use a picture in your answer and label it.)

5. You run program A using kernel threads, and then re-run it using user-level threads. How could these two runs behave differently with respect to load and store instructions?

6. Draw the structure of a standard 32-bit virtual address with 4K pages. What bad things happen if you switch the order of the two components?

7. Your (ancient) machine has an 8K, direct mapped, physical cache with 4K pages. Program A is using an 8K, page-aligned array, whose first page maps to physical page 31. Give the set of physical pages that the second page of the array should be mapped to and why.

8. What address space layout will be the best for a linear page table as compared a hashed page table and vice versa? (Make sure to say why.)

9. Give the simplest example of a chunk of data and a predictable access pattern that LRU will perform optimally bad on. In the absence of prefetching, what is the best *realistic* algorithm to use?

10. Joebob, your mortal enemy, gives you a USB stick that you want to mount as a file system on your computer. Give the type of checks that the file system should do before treating the USB data as a valid file system.

11. You create a file on a Unix file system (such as FFS). Roughly: what meta data do you modify, what errors could you get, what order do you write the metadata out in?

12. Among NFS's operations are:

```
// write nbytes from buf into offset of the file named by fh
write(fh, offset, buf, nbytes)

// make directory "name" in directory named by dh
mkdir(filehandle, name)
```

It sends these requests across a lossy network, so may obviously have to retransmit them. Discuss: what problems could occur because of retransmission for these two operations and a (partial) fix.

13. You have a distributed file system. What is the perfect guarantee it could give for cache consistency? What would you have to do to implement this? Is there any way for an application running on top of this perfect consistency could see stale data?

14. Many distributed file systems implement close-to-open consistency. Give an intuitive statement of what this is, and two reasons two reasons you might prefer it over perfect consistency (including one non-performance reason)

15. Let' say you have a network interrupt handler that looks something like:

```
interrupt_handler() {

        while(there are packets to receive)
                pull pkt off network interface
                enqueue(pkt);

        while(there are packets to transmit)
                dequeue from transmit queue
                give to network interface
}
```

You notice that sometimes no packets come out of the system for awhile, in situations where they should. Similarly, you notice that sometimes applications do not run, but should. What problems in this code could cause this behavior? Give a sketch of how to fix it.

Systems Software Comprehensive Exam

Fall 2005

Solutions by 2006 First Years

1. Write the shortest non-reentrant legal C function you can and why you think it is non-reentrant.

   Solution:

```
int foo() {
     static int x;
     x++;
     return x;
}
```

   This function is not reentrant because it modifies a global (static) variable before returning it. Calling it at different times from different threads will alter the variable x and return different results.

2. Your machine can only do 32-bit loads and stores. What is the locking-related problem in the following code? How would you fix it?

```
struct foo {
     lock_t a_lock; // always held before touching a
     lock_t b_lock; // always held before touching b

     char a;
     char b;
};
...

void update_ab(struct foo *f) {
     lock(f->a_lock);
     f->a++;
     unlock(f->a_lock);

     lock(f->b_lock);
     f->b++;
     unlock(f->b-lock);
}
```

   Solution: The fields `a` and `b` in `struct foo` are both `char` variables that are 1-byte in length, so the compiler will place them together within one 4-byte (32-bit) chunk in memory. Because we can only read/write 32-bits at a time, when we do `f->a++`, we actually have to read the 32-bit chunk containing both the values of `a` and `b`,

modify the 1 byte corresponding to `a` by incrementing it, and then write back the entire 32-bit chunk. If another thread attempts to modify `f->b` and that operation finishes before the original thread's call to `f->a++`, then when the new value of `a` is written back to memory, the ORIGINAL value of `b` is also written back, thus clobbering the new `b` value. This problem can be solved by adding padding in the `struct` to ensure that `a` and `b` are in separate 32-bit chunks. e.g.,

```
struct foo {
     lock_t a_lock;
     lock_t b_lock;
     char a;
     char padding[3];
     char b;
}
```

3. Your threaded code has no race conditions. It does have this routine:

```
foo() {
     lock(a);
     lock(b);
     ...
     unlock(a);
     ...
     lock(a);
     ...
     unlock(a);
     unlock(b);
}
```

Ignore performance: what is wrong here?

Solution: This code can deadlock as follows.

Thread 1:	Thread 2:
lock(a)	-
lock(b)	-
unlock(a)	-
	lock(a)
lock(a) WAITING	-
-	lock(b) WAITING

Deadlock!

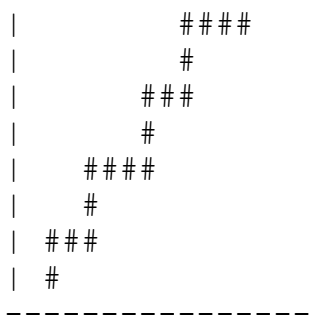
4. In what way is a good proportional-share process scheduling algorithm essentially equivalent to a good graphics line-drawing algorithm? (Use a picture in your answer and label it.)

Solution: In a proportional-share process scheduling algorithm (otherwise known as a lottery system), each process has a priority specified by a percent, where all

currently-active processes have percents that add up to 100%. Each process gets a number of tickets proportional to its priority (say tickets numbered between 1 to 100, so if a process has 20% priority, it might receive 20 tickets). During each scheduling quantum, the scheduler picks a random number, and whichever process owns the ticket for the picked number gets to run. Thus, on average, processes with more tickets (higher priorities) run more than those with less tickets (lower priorities).

In a graphics line-drawing algorithm, the challenge is to draw a continuous line (in x dimensions, let's say 2 for simplicity) by drawing discrete pixels in a pattern such that when one steps back, the pixels form a line with a particular slope.

What the two have in common is taking a continuous process and mapping it into the discrete domain.



Imagine that there are only 2 processes. On one axis is the amount of time that one process runs and on the other axis is the amount of time that the other process runs. Given enough samples (and a really long run), this will look like a STRAIGHT LINE with slope determined by the relative priorities of the 2 processes (a slope of 1 if each process has 50% chance of running at any given scheduling quantum). However, the scheduling world is discrete, so there will be chunks of time (demonstrated by the '#' in my pathetic ASCII art) where one process will run and then the other process (or maybe the same process) will run. This sort of draws a line, albeit stochastically.

5. You run program A using kernel threads, and then re-run it using user-level threads. How could these two runs behave differently with respect to load and store instructions?

Solution: If you're running your program on one processor, then all the loads/stores across different threads are guaranteed to be atomic/sequential. However, kernel threads can run simultaneously on multiple processors, so loads/stores do not have this sequential consistency guarantees. User-level threads can only run on one processor, so you still have the sequential consistency guarantees.

Another answer we came up with: Loads/stores can can cause page faults, so with kernel threads, the kernel knows about page faults and only blocks the 1 thread with the faulting instruction, but for user-level threads, if one thread blocks on a page fault, the entire process blocks because from the kernel's point-of-view, there is only one thread.

6. Draw the structure of a standard 32-bit virtual address with 4K pages. What bad things happen if you switch the order of the two components?

   Solution:
   [20 bits for virtual page number (VPN)][12 bits for page offset]
   Backwards:
   [12 bits for page offset][20 bits for virtual page number (VPN)]

   The reason why the least significant bits are assigned to the page offset is that you often access nearby bytes together (spatial locality), so you want those to map to the same page. If you get the order backwards and instead assign the least significant bits to the VPN, then you lose spatial locality (locality of reference) and you get horrendous performance because every time you access nearby bytes in memory, you end up accessing completely different pages, thus killing your TLB hit rate and causing other slowdowns.

7. Your (ancient) machine has an 8K, direct mapped, physical cache with 4K pages. Program A is using an 8K, page-aligned array, whose first page maps to physical page 31. Give the set of physical pages that the second page of the array should be mapped to and why.

   Solution: Any even-numbered page would work.
   There are two parts to this problem:

   - First, you have an 8K direct-mapped cache reference by physical addresses. 8K = 2^{13}, so here's how 32-bit addresses map to the cache:
   `[ 19 bits (cache tag) ][ 13 bits (cache index) ]`

   - Second, you have a virtual memory system with 4K pages, so here's how 32-bit addresses map to the virtual memory (recall that $4K=2^{12}$)
   `[ 20 bits (page number) ][ 12 bits (page offset) ]`

   You have an 8K, page-aligned array whose first page maps to physical page 31. This first page contains the first half of the array, and the second half of the array fits entirely on some other page. So where does this array reside in physical memory? Let's dissect the page number 31:

   This is the address range of the first half of the array:

```
[ 20 bits (page number) ][ 12 bits (page offset) ]
0000 0000 0000 0001 1111  0000 0000 0000
0000 0000 0000 0001 1111  FFFF FFFF FFFF
```

Now where does this array fit into our 8K direct-mapped cache? Well, let's see:

```
[ 19 bits (cache tag)  ][ 13 bits (cache index) ]
0000 0000 0000 0001 111 1 0000 0000 0000
0000 0000 0000 0001 111 1 FFFF FFFF FFFF
```

Notice that the cache tag is `0b1111`, and the cache indices range from: `1 0000 0000 0000` to `1 FFFF FFFF FFFF`. This is the 4K that comprises the upper-half of the 8K cache. Now, in order to maximize cache hits, we want to put the second-half of the array in the LOWER-HALF of the 8K cache. How can we do that? By ensuring that the MSB (most significant bit) of the cache index is a 0. What does that mean in terms of the physical page number? It must end in 0.

```
Cache:  [ 19 bits (cache tag) ][ 13 bits (cache index) ]
        ???? ???? ???? ???? ??? 0 0000 0000 0000
        ???? ???? ???? ???? ??? 0 FFFF FFFF FFFF
VM:     [ 20 bits (page number)  ][12 bits (page offset)]
```

It doesn't matter what the rest of the physical page number is, as long as it ends in a 0, which means that it must be EVEN.

8. What address space layout will be the best for a linear page table as compared to a hashed page table and vice versa? (Make sure to say why.)

   Solution: Linear page table is better for densely-filled memory because a hashed page table will have lots of collisions with a densely-filled memory. Hashed page table is good for sparsely-filled memory because a linear page table will waste more space with a sparsely-filled memory.

9. Give the simplest example of a chunk of data and a predictable access pattern that LRU will perform optimally bad on. In the absence of prefetching, what is the best *realistic* algorithm to use?

   Solution: LRU is terrible whenever you have a working set that is 1 larger than the size of your cache, and you're accessing the elements repeatedly in a loop.

   Example, 3 addresses (0, 1, 2) and size-2 cache:
   0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, etc... Every single access is a cache miss!

   Random replacement is the best realistic algorithm to use.

10. Joebob, your mortal enemy, gives you a USB stick that you want to mount as a file system on your computer. Give the type of checks that the file system should do before treating the USB data as a valid file system.

    Solution: You should check that the FS is well-formed - e.g., no cycles in directory structure, the . and .. entries point to right places, link counts are correct, etc.

11. You create a file on a Unix file system (such as FFS). Roughly: what meta data do you modify, what errors could you get, what order do you write the metadata out in?

    Solution: Metadata modified: inodes, inode reference count, timestamp, pointers to data blocks, directory entry

    Possible errors: out of disk space, out of inodes, permission errors, hardware errors

    Order to write metadata out to disk:
    1.) Create and initialize data blocks
    2.) Allocate and initialize inode to point to data blocks
    3.) Add the filename to the directory entry and point it to the appropriate inode

    You want to do things in this order because if you crash sometime in the middle, you won't be left with a pointer to garbage. General rule: always initialize something before setting a pointer to it.

12. Among NFS's operations are:

```
//write nbytes from buf into offset of the file named by fh
write(fh, offset, buf, nbytes)

//make directory "name" in directory named by dh
mkdir(dh, name)
```

    It sends these requests across a lossy network, so may obviously have to retransmit them. Discuss: what problems could occur because of retransmission for these two operations and a (partial) fix.

    Solution: If you do 2 successive writes of different contents to the same file, the two write commands could arrive out-of-order if the network is slow and retransimission is necessary. This could be a problem if, say, you did write(file, "FOO") and then write(file, "BAR"). After the second write, you might expect that the file contains "BAR", but if they arrive out-of-order, then the "FOO" write might come after the "BAR" write. Similar problem with mkdir.

    One partial fix is to put some sort of sequence numbers to ensure in-order delivery.

    (We're not sure whether NFS guarantees in-order delivery even in the presence of

retramissions, etc.)

13. You have a distributed file system. What is the perfect guarantee it could give for cache consistency? What would you have to do to implement this? Is there any way an application running on top of this perfect consistency could see stale data?

    Solution: Perfect cache consistency guarantee is called 'write-to-read' consistency, which means that when one client reads a file, it should have the latest contents of the last thing that anybody else ever wrote to the file. You would need to lock files a lot in order to implement this and update all local clients' caches whenever there is a write to a file by one client.

    We were unsure about if there was any way that an app running on top of perfect consistency could see stale data:

    Yes - If the network is slow, then the clients' cache updates might not happen fast enough so that clients can still see stale data.

    No - by definition, it's perfect.

14. Many distributed file systems implement close-to-open consistency. Give an intuitive statement of what this is, and two reasons you might prefer it over perfect consistency (including one non-performance reason).

    Solution: Close-to-open consistency means that whenever a client opens a file, he is guaranteed to at least see the data on the file at the time when it was last closed. This is a weaker guarantee than 'write-to-read' consistency because another client may have that same file opened and be writing to it.

    Advantages:
    Performance - You no longer have to invalidate/update everyone's cache whenever you do a write or use that many locks, etc. You just need to do updates when you close a file, which is much less frequent.

    Robustness - If you're writing to a file and then crash or have your data corrupted without closing the file, other clients don't see your corrupted version. (This isn't a great reason, so somebody could probably think of a better one ...)

15. Let's say you have a network interrupt handler that looks something like:

```
interrupt_handler() {
     while(there are packets to receive)
          pull pkt off network interface
          enqueue(pkt);
     while(there are packets to transmit)
```

```
            dequeue from transmit queue
            give to network interface
}
```

You notice that sometimes no packets come out of the system for awhile, in situations where they should. Similarly, you notice that sometimes applications do not run, but should. What problems in this code could cause this behavior? Give a sketch of how to fix it.

Solution: No packets coming out of system for a while, when there are packets to transmit - This is a case of starvation and can occur whenever there are a constant stream of packets to receive. The interrupt handler keeps on handling received packets and never gets to send packets.

Applications don't run - The applications can suffer from starvation when all the time is spent in the interrupt handler. This can occur whenever there is a constant stream of packets being sent or received.

The problem is that the interrupt handler does not give up control as long as there are packets to receive and/or transmit, and there is no way to interrupt the interrupt handler (because interrupts are run with interrupts disabled).

One solution might be to modify the code so that 2 things happen:
1.) The transmit loop has a chance of running even if lots of packets are being received
2.) The application has a chance of running even if lots of packets are being sent or received

One possibility is a lottery system where, in the interrupt handler, the receive handler has 1/3 chance of running, the transmit handler has 1/3 chance of running, and the application itself has 1/3 chance of running. Then as the program progresses, dynamically change those percentages to adapt to the rate at which the different handlers and application execute, providing a kind of negative feedback. A simpler scheme would be to put a limit of how many packets to process in a particular time range.