AA

Comprehensive Exam: Algorithms and Concrete Mathematics Autumn 2000

This is a one hour closed-book exam and the point total for all questions is 60.
All of the intended answers may be written within the space provided. If necessary, you may use the back of the preceding page for additional 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 part in seeing to it that others as well as themselves uphold the Honor Code.*

B. *The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and reasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic pressures which may lead to violation of the Honor Code.*

C. *While the faculty alone has the responsibility for setting 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	# 5	Total
Score						
Max	10	20	10	10	10	60

1. (*10 points*) *Big-Oh notation, Running times*
For each of the following functions, circle the best upper bound from among the choices given. You only need to circle one answer in each case. For example, if you say that a function is $O(n)$, you do not need to also say that it is also $O(n^2)$, $O(n \log n)$, and so on.

(a) $f(n) = \sqrt{n}$
Choose: $O(n)$ $O(n^2)$ $O(n^3)$ $O(\log n)$ $O(n \log n)$ $O(2^n)$ none

(b) $f(n) = (\sqrt{n}+1)(\sqrt{n}-1)\sqrt{n}$
Choose: $O(n)$ $O(n^2)$ $O(n^3)$ $O(\log n)$ $O(n \log n)$ $O(2^n)$ none

(c) $f(n) = 422n^3 + 5n^2 \log n + 121231234$
Choose: $O(n)$ $O(n^2)$ $O(n^3)$ $O(\log n)$ $O(n \log n)$ $O(2^n)$ none

(d) $f(n) = n^n$
Choose: $O(n)$ $O(n^2)$ $O(n^3)$ $O(\log n)$ $O(n \log n)$ $O(2^n)$ none

For each of the following programs (or fragments thereof) give a good upper bound on the running time of the algorithm using "big-Oh" notation, as a function of the value of n.

(e) (*2 points*)

```
for i from 1 to n do
    for j from i to n do
        ⟨ something that is O(log n) ⟩
```

(f) (*2 points*)

```
/* print n in binary if n > 1; mod runs in constant time */
bprint(n)
    if n > 0 then
        bprint(⌊n/2⌋);
        print(n mod 2);
```


2

2. (*20 points*) *Recurrences.*
Solve each of these three recurrences. You may give an exact solution, or give a good upper-bound using big-oh notation.

a) If $f(n) = f(n/2) + \log n$, with $f(1) = 0$, then find a closed form expression for $f(n)$.

b) If $g(n) = 2g(n/2) + \sqrt{(}n)$, with $g(0) = 0$ and $g(1) = 1$, then find a closed form expression for $g(n)$.

c) If $h(n) = h(n/2) + h(n/4)$, with $h(0) = h(1) = 1$, then find a closed-form expression for $h(n)$.

4

3. (*10 points*) *Matching.*
One day, upon returning from the laundromat, I realize that, as usual, I've lost a few socks. In fact, no two of my socks are an exact match! Luckily, though, some of them are close and I have a similarity scoring function: any two socks i and j have an associated score $0 \leq s_{ij} \leq 1$ which is large if they are very similar and small if they are very different. Naturally, $s_{ij} = s_{ji}$ for all i and j. I have an even number of socks and decide to try to put my socks together two-by-two so as to maximize the sum of scores.

I decide to use the following greedy algorithm:

Repeat until all socks have mates:

- Pick a random unmated sock i.
- Among the other unmated socks, find the sock j so that s_{ij} is maximum (i.e., no other unmated sock k has $s_{ik} > s_{ij}$).

Show that this method may produce a poor matching. More specifically, the *score* for a matching is the sum of the similarty scores for the pairs I pick. Show that for any $0 < \delta < 1$, there is a set of socks, and similarity scoring function as described above, so that if I pick socks in a particular order I will get a score less than δ times the score for the best overall matching of this set of socks.

4. (*10 points*) *Amortized Analysis.*
A bank offers a combined savings/checking account with options to deposit \$1000 to the savings account, deposit \$1000 to checking, withdraw an integer multiple of \$1000 from the savings account, withdraw an integer multiple of \$1000 from checking, or transfer an integer multiple of \$1000 from the savings to the checking account. Each of these operations has a certain overhead cost to the bank, but actually storing the money costs them nothing. (In the event that you attempt to withdraw more money from an account than you actually have in that account, or to transfer more money from savings than you have in savings, the operations fails and costs nothing to the bank.) Below is a table of the operations and the overhead costs to the bank for each of them (assume α and β are constants):

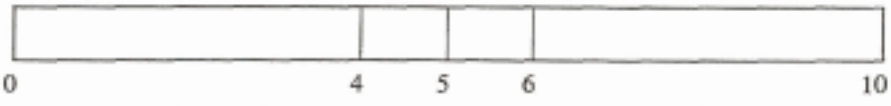
Option	Actual Cost to the Bank
Deposit \$1000 to Savings	α
Deposit \$1000 to Checking	α
Withdraw $k \times$ \$1000 from savings	$\alpha + k\beta$
Withdraw $k \times$ \$1000 from checking	$\alpha + k\beta$
Transfer $k \times$ \$1000 from savings to checking	$\alpha + k\beta$

(a) Say you open an account today with a balance of \$0. Show that the bank can offset the overhead costs from your first n operations by simply charging an $O(1)$ service fee every time you deposit to savings or checking. That is, give an amount the bank could charge so that, regardless of your first n operations, the overhead would not exceed the total service charge, and show why this fee would offset the overhead costs.

6

(b) Now say the bank adds a sixth option: "transfer an integer multiple of \$1000 from *checking* to *savings*," and assume that the bank decides it will still only charge a service fee for deposits, and that this fee will still be constant. Show that, no matter what the service fee is, you could open a new account (also with an initial balance of \$0) and perform a series of operations so that the bank would lose money.

(c) *(10 points) Dynamic Programming.*
Assume you have an l-foot log (i.e., a tree trunk, not a logarithm) that has a few marks spraypainted on it, indicating that you need to saw it at these places. For example, the log may look like the one below:



where the numbers indicate distance from the left end of the log. Assume further that the cost of making a particular cut is the length of the section in which you make the cut. For example, in the diagram, if we cut first at 4, then at 5, then at 6, the cost is $10 + 6 + 5$.

Give an efficient, dynamic-programming algorithm for deciding the cheapest cut order. More precisely, assume you are given a list $L = (l_1, l_2, ..., l_k)$ of cuts, where the ith cut l_i is given as the distance between the left end of the log and the place at which that cut is to be made. (The above log would have the list $L = (4, 5, 6)$.) Your algorithm should take such a list and output an ordered list L' that gives the cuts in the cheapest order.

Also, give a big-Oh time bound for your algorithm.

SOLUTIONS AA

Comprehensive Exam: Algorithms and Concrete Mathematics Autumn 2000

1. (*10 points*) *Big-Oh notation, Running times*
For each of the following functions, circle the best upper bound from among the choices given. You only need to circle one answer in each case. For example, if you say that a function is $O(n)$, you do not need to also say that it is also $O(n^2)$, $O(n \log n)$, and so on.

   (a) (*1 points*) $f(n) = \sqrt{n}$
   Choose: $O(n)$ $O(n^2)$ $O(n^3)$ $O(\log n)$ $O(n \log n)$ $O(2^n)$ none
   Answer: $O(n)$

   (b) (*1 points*) $f(n) = (\sqrt{n}+1)(\sqrt{n}-1)\sqrt{n}$
   Choose: $O(n)$ $O(n^2)$ $O(n^3)$ $O(\log n)$ $O(n \log n)$ $O(2^n)$ none
   Answer: $O(n^2)$

   (c) (*2 points*) $f(n) = 422n^3 + 5n^2 \log n + 121231234$
   Choose: $O(n)$ $O(n^2)$ $O(n^3)$ $O(\log n)$ $O(n \log n)$ $O(2^n)$ none
   Answer: $O(n^3)$

   (d) (*2 points*) $f(n) = n^n$
   Choose: $O(n)$ $O(n^2)$ $O(n^3)$ $O(\log n)$ $O(n \log n)$ $O(2^n)$ none
   Answer: None of the above.

   For each of the following programs (or fragments thereof) give a good upper bound on the running time of the algorithm using "big-Oh" notation, as a function of the value of n.

   (e) (*2 points*)

   for i **from** 1 **to** n **do**
   for j **from** i **to** n **do**
   ⟨ something that is $O(\log n)$ ⟩

   Answer: $O(n^2 \log n)$

   (f) (*2 points*)

   /* print n in binary if $n > 1$; *mod* runs in constant time */
   bprint(n)
   if $n > 0$ **then**
   bprint($\lfloor n/2 \rfloor$);
   print(n mod 2);

Answer: $O(\log n)$

2. (*20 points*) *Recurrences.*
Solve each of these three recurrences. You may give an exact solution, or give a good upper-bound using big-oh notation.

   (a) (*5 points*) If $f(n) = f(n/2) + \log n$, with $f(1) = 0$, then find a closed form expression for $f(n)$.
   Answer: $f(n) = (\log^2 n + \log n)/2 = O(\log^2 n)$

   (b) (*5 points*) If $g(n) = 2g(n/2) + \sqrt{(n)}$, with $g(0) = 0$ and $g(1) = 1$, then find a closed form expression for $g(n)$.
   Answer: $g(n) = \frac{\sqrt{2}+2}{3}n - \frac{\sqrt{2}+1}{3}\sqrt{n} = O(n)$

   (c) (*10 points*) If $h(n) = h(n/2) + h(n/4)$, with $h(0) = h(1) = 1$, then find a closed-form expression for $h(n)$.
   Answer: $h(n) = F_{\log n}$ where F_n is the nth Fibonacci number.

3. (*10 points*) *Matching.*
One day, upon returning from the laundromat, I realize that, as usual, I've lost a few socks. In fact, no two of my socks are an exact match! Luckily, though, some of them are close and I have a similarity scoring function: any two socks i and j have an associated score $0 \leq s_{ij} \leq 1$ which is large if they are very similar and small if they are very different. Naturally, $s_{ij} = s_{ji}$ for all i and j. I have an even number of socks and decide to try to put my socks together two-by-two so as to maximize the sum of scores.

   I decide to use the following greedy algorithm:

   Repeat until all socks have mates:

   - Pick a random unmated sock i.
   - Among the other unmated socks, find the sock j so that s_{ij} is maximum (i.e., no other unmated sock k has $s_{ik} > s_{ij}$).

   Show that this method may produce a poor matching. More specifically, the *score* for a matching is the sum of the similarty scores for the pairs I pick. Show that for any $0 < \delta < 1$, there is a set of socks, and similarity scoring function as described above, so that if I pick socks in a particular order I will get a score less than δ times the score for the best overall matching of this set of socks.

   Answer: Consider four socks, a, b, c, and d, with $s_{ab} = 1$, $s_{bc} = \frac{\delta}{2}$, and all other scores equal to 0. If the first sock I randomly pick is c, I will match it with b for a score of less than δ, where the best possible score (from putting a with b and c with d) would be 1.

4. (*10 points*) *Amortized Analysis.*
A bank offers a combined savings/checking account with options to deposit \$1000 to the savings account, deposit \$1000 to checking, withdraw an integer multiple of \$1000 from the savings account, withdraw an integer multiple of \$1000 from checking, or transfer an integer multiple of \$1000 from the savings to the checking account. Each of these operations has a certain overhead cost to the bank, but actually storing the money costs them nothing. (In the event that you attempt to withdraw more money from an account than you actually have in that account, or to transfer more money from savings than you have in savings, the operations fails and costs nothing to the bank.) Below is a table of the operations and the overhead costs to the bank for each of them (assume α and β are constants):

Option	Actual Cost to the Bank
Deposit \$1000 to Savings	α
Deposit \$1000 to Checking	α
Withdraw $k \times$ \$1000 from savings	$\alpha + k\beta$
Withdraw $k \times$ \$1000 from checking	$\alpha + k\beta$
Transfer $k \times$ \$1000 from savings to checking	$\alpha + k\beta$

(a) (*5 points*) Say you open an account today with a balance of \$0. Show that the bank can offset the overhead costs from your first n operations by simply charging an $O(1)$ service fee every time you deposit to savings or checking. That is, give an amount the bank could charge so that, regardless of your first n operations, the overhead would not exceed the total service charge, and show why this fee would offset the overhead costs.

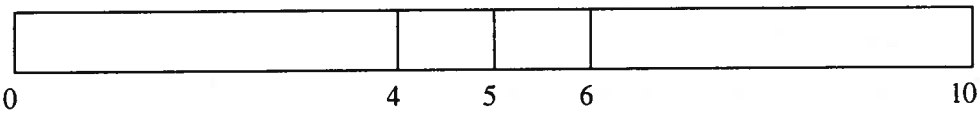
Answer: Amortized cost of $3\alpha+2\beta)$ for each deposit to savings, and of $2(\alpha+\beta)$ for each deposit to checking. Every thousand dollars deposited to savings costs $\alpha + \beta$ exactly, and then can cost at most twice this much again, if it is transferred to checking and then withdrawn (if it is withdrawn directly from savings, or if it is transferred or withdrawn in along with more money, the individual cost of moving just this thousand dollars is even less than $2(\alpha + \beta)$). Therefore the cost given is sufficient. A similar argument can be made for deposits to checking.

(b) (*5 points*) Now say the bank adds a sixth option: "transfer an integer multiple of \$1000 from *checking* to *savings*," and assume that the bank decides it will still only charge a service fee for deposits, and that this fee will still be constant. Show that, no matter what the service fee is, you could open a new account (also with an initial balance of \$0) and perform a series of operations so that the bank would lose money.

Answer: Assume the bank applies a constant fee of l for each deposit to savings, and of m for each deposit to checking. Let $k = \max(l, m)$. Simply deposit one

thousand dollars into savings and then transfer it back and forth over an over. If you transfer it back and forth $\frac{2k}{\alpha+\beta}$ times (rounding up to the nearest integer, of course), you will cost the bank $2k$.

5. (*10 points*) *Dynamic Programming.*
Assume you have an l-foot log (i.e., a tree trunk, not a logarithm) that has a few marks spraypainted on it, indicating that you need to saw it at these places. For example, the log may look like the one below:



where the numbers indicate distance from the left end of the log. Assume further that the cost of making a particular cut is the length of the section in which you make the cut. For example, in the diagram, if we cut first at 4, then at 5, then at 6, the cost is $10 + 6 + 5$.

Give an efficient, dynamic-programming algorithm for deciding the cheapest cut order. More precisely, assume you are given a list $L = (l_1, l_2, ..., l_k)$ of cuts, where the ith cut l_i is given as the distance between the left end of the log and the place at which that cut is to be made. (The above log would have the list $L = (4, 5, 6)$.) Your algorithm should take such a list and output an ordered list L' that gives the cuts in the cheapest order.

Also, give a big-Oh time bound for your algorithm.

Answer: First, let us define $l_0 = 0$ and $l_{k+1} = l$. We will have a $k+1 \times k+1$ array C where the ijth entry, c_{ij} gives the cheapest cost of making the cuts *between* l_i and l_j (not including the ith and jth cuts), when $i \leq j$. Initialize $c_{ii} = 0$, for all $0 \leq i \leq k+1$, $c_{i,i+1} = 0$, for all $0 \leq i \leq k$, $c_{i,i+2} = l_{i+2} - l_i$, for all $0 \leq i \leq k-1$. (If you care, set to zero all entries c_{ij} where $i > j$.)

Next, we will have another array D of the same dimensions, where the ijth entry d_{ij} gives the first cut we should make between l_i and l_j, given that those two have been made already. Initialize this array to have all -1s, if you like.

Now, for each g in $3 \ldots k+1$, letting i range from 0 to $k+1-g$, find $d_{i,i+g}$. How do we do this? Well, $c_{i,i+g} = l_{i+g} - l_i + \min_{i<h<i+g} c_{ih} + c_{h,i+g}$, so, using the array as a lookup table, find the best value of h for this particular $c_{i,i+g}$, and set $c_{i,i+g}$ and $d_{i,i+g}$ accordingly. At the end, $d_{0,k+1}$ will have the first cut, say l'_1, that you should make, and you can figure out the next two cuts as d_{0,l'_1} and $d_{l'_1,k}$, and so on.

This algorithm runs in time $O(k^2)$.

13

AI

Computer Science Department
Stanford University
Comprehensive Examination in Artificial Intelligence
Autumn 2000

October 31, 2000

PLEASE READ THIS FIRST

a. You should write your answers for this part of the Comprehensive Examination in a BLUE BOOK. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.

b. Be sure you have all the pages of this exam. There are 4 pages in addition to this cover sheet.

c. This exam is OPEN BOOK. You may use notes, articles, or books – but no help from people or computers.

d. Show your work, since PARTIAL CREDIT will be given for incomplete answers. For example, you can get credit for making a reasonable start on a problem even if the idea or arithmetic does not work out. You can also get credit for realizing that certain approaches are incorrect.

e. Points in this exam add up to 60. Points are allocated according to the number of minutes we believe a student familiar with the material should take to answer the questions. If you are somewhat less familiar with the material, a question may easily take you longer than the number of points it's worth. Therefore be careful: **IF YOU ARE TAKING TOO LONG ON A QUESTION, WRITE DOWN WHATEVER YOU HAVE AND MOVE ON.**

1 Search (8 points)

a. Uninformed Search

(i) (1 point) Describe or give an example of a search space where depth-first search will perform much better than iterative deepening search

(ii) (1 point) Describe or give an example of a search space where breadth-first search will perform much better than depth-first search

(iii) (1 point) Describe or give an example of a search space where depth-first search will perform much better than breadth-first search

b. Heuristic Search

(5 points) A* search involves evaluating search paths via $\hat{f} = g + \hat{h}$, where g is the lowest cost path to the current search state, and $\hat{h}$ is the heuristic function which provides an optimistic estimate of the cost to a goal state. Now assume that the heuristic function $\hat{h}$ is induced from a cost function between nodes $h'(x, y)$ which obeys the triangle inequality. (The triangle inequality says that the sum of the costs from x to z and z to y must not be less than the cost from x to y directly.) Prove that the $\hat{f}$-cost along any path in the search tree is nondecreasing.

2 Logic: resolution (12 points)

(This question comes from (the late) Jon Barwise and (our new provost) John Etchemendy. It also appears in the exercises of Russell and Norvig.)

Consider the following statements:

> If a unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical, if it is horned.

From the above, can you prove that the unicorn is mythical? How about magical? Horned? Use resolution for your proofs (using propositional logic is possible and acceptable). Show:

a. (2 points) the basic logical translation of this text

b. (2 points) the translations of these into a form suitable for resolution theorem proving

c. (8 points) For each of the three unicorn properties (mythical, magical, horned), if it can be proved, show your proof. If it can't be proved, explain the justification for being able to conclude that the fact that unicorns have this property doesn't follow from the information given.

3 Probabilistic models (12 points)

a. (1 point) A scientist tells us that 1/3 of all kangaroos have blue eyes, and 1/3 of all kangaroos are left-handed. On the basis of this information alone, what bounds can we place on the proportion of kangaroos that are both blue-eyed and left-handed?

b. (2 points) A loglinear model for the joint distribution of some number of categorical variables $X_1, \ldots, X_n$ takes the following general form:

$$\log P(X_1 = x_1, \ldots, X_n = x_n) = \sum_{C \in \mathcal{C}} \lambda_C(x_C)$$

where each $C \subseteq \{1, \ldots, n\}$ (so $\mathcal{C} \subseteq 2^{\{1,\ldots,n\}}$). That is, x_C is some subset of the x_i variables, and $\lambda_C(\cdot)$ is a function of this subset.

Show that all Bayesian networks ("graphical models") are loglinear models.

c. (9 points) Consider this situation: You saw John talking to his boss. Later you saw John looking upset. John may have been upset because his boss gave him a warning. Or he may be upset because his boyfriend left him.

(i) (1 point) Draw a suitable Bayesian network to represent the causal structure among the 4 random variables B for boss talking to John, W for boss warning John, U for John being upset, and L for his boyfriend leaving him.

(ii) (1 point) Make up conditional probability tables for each node. [HINT: keep the numbers simple. Use quarters, thirds, and halves!!]

(iii) (4 points) Given the information above (B and U), calculate the probability that John's boss warned him.

(iv) (3 points) Calculate the probability that John's boss warned him given the above information and that you know that his boyfriend left him.

Your answers may be approximate, but you should show your work

4 Learning (14 points)

a. (2 points) A circuit has two input values A and B whose values are either +1 for 'true' or −1 for 'false'. Design a perceptron (Linear Threshold Unit) network which computes the function not (A or B). Draw the network and indicate clearly all weights and threshold values (assume the network outputs 1 if the dotproduct of the weights and the inputs is greater than some threshold t specified by each unit).

b. (3 points) We are trying to predict whether Comps questions are easy or difficult ($D = +$ if they are difficult) based on two features:

L Whether they are long (1 = long)

M Whether they have a lot of math (1 = yes)

For training data, we have examined 12 Comps questions, and have collected the following statistics, which we show twice: on the left are counts for the diferent data patterns, and on the right the data is shown in a contingency table showing $-:+$ counts for each combination of the classificatory variables.

L	M	D	count
0	0	$-$	4
0	0	$+$	1
0	1	$-$	0
0	1	$+$	3
1	0	$-$	1
1	0	$+$	2
1	1	$-$	1
1	1	$+$	0

		M	
		0	1
L	0	4:1	0:3
	1	1:2	1:0

Draw a decision tree for this data (using information gain for node splitting, and no stopping criterion or pruning, so that the tree is grown so long as there is some classificatory feature that appears to have information about the target feature).

c. (4 points) A Naive Bayes classifier for this problem predicts the target feature from the prior and independently from the classificatory features as follows:

Choose $\hat{d} = \arg\max_{d=\{-,+\}} P(d)P(L = l|d)P(M = m|d)$

(That is, it calculates the expression shown for both $d = +$ and $d = -$ and chooses the value of d that gives the expression higher probability.) What classificatory decisions would a Naive Bayes classifier make for each combination of classificatory variables?

d. (3 points) Suppose we have 3 test data instances, whose correct classification we know, as follows:

L	M	D
0	0	$-$
1	1	$+$
0	1	$+$

What is the decision of each classifier on each datum? Which does better overall?

e. (2 points) Is the better performance of one learner reasonable or surprising here?

5 Vision and Natural Language Processing (14 points)

a. (2 points) One can easily use a segmentation algorithm to find edges in images. However, it is generally hard to use an edge detector to segment an image into regions. Why doesn't the output of an edge detector segment an image into regions?

b. (2 points) You want to design a vision system that uses shading information to recover the three-dimensional geometry of the scene from pictures taken in the real world. You don't know what will be in front of the camera. What key difficulties are you going to face?

c. (10 points) Write:

(i) (1 point) a set of phrase structure grammar rules

(ii) (1 point) a lexicon

(iii) (3 points) a set of meanings for words in the lexicon, and

(iv) (3 points) semantic combination rules for the grammar rules, in the form of a definite clause grammar

for the sentence:

Sydney is a beautiful city.

You can assume that the adjective *beautiful* has intersective semantics, and that *Sydney* is a constant – that is, a suitable semantic form for the sentence is: beautiful(sydney) & city(sydney). You may find it helpful to look at, but will need to extend, the grammar of Russell & Norvig pp. 665ff. In particular, the semantic forms in your lexicon will need to include lambda-expressions, as shown there.

(v) (2 points) Discuss the most salient difficulties involving the semantics of this sentence and your grammar for it which would make extending your grammar difficult.

AI
SOLUTIONS

Computer Science Department
Stanford University
Comprehensive Examination in Artificial Intelligence
Autumn 2000

October 31, 2000

Solution Samples

PLEASE READ THIS FIRST

a. You should write your answers for this part of the Comprehensive Examination in a BLUE BOOK. Be sure to write your MAGIC NUMBER on the cover of every blue book that you use.

b. Be sure you have all the pages of this exam. There are 9 pages in addition to this cover sheet.

c. This exam is OPEN BOOK. You may use notes, articles, or books – but no help from people or computers.

d. Show your work, since PARTIAL CREDIT will be given for incomplete answers. For example, you can get credit for making a reasonable start on a problem even if the idea or arithmetic does not work out. You can also get credit for realizing that certain approaches are incorrect.

e. Points in this exam add up to 60. Points are allocated according to the number of minutes we believe a student familiar with the material should take to answer the questions. If you are somewhat less familiar with the material, a question may easily take you longer than the number of points it's worth. Therefore be careful: **IF YOU ARE TAKING TOO LONG ON A QUESTION, WRITE DOWN WHATEVER YOU HAVE AND MOVE ON.**

1 Search (8 points)

a. Uninformed Search

(i) (1 point) Describe or give an example of a search space where depth-first search will perform much better than iterative deepening search

(ii) (1 point) Describe or give an example of a search space where breadth-first search will perform much better than depth-first search

(iii) (1 point) Describe or give an example of a search space where depth-first search will perform much better than breadth-first search

b. Heuristic Search

(5 points) A* search involves evaluating search paths via $\hat{f} = g + \hat{h}$, where g is the lowest cost path to the current search state, and $\hat{h}$ is the heuristic function for the cost to a goal state. Now assume that the heuristic function $\hat{h}$ is induced from a function between nodes $h'(x, y)$ which provides an optimistic estimate of the cost and which obeys the triangle inequality. (The triangle inequality says that the sum of the costs from x to z and z to y must not be less than the cost from x to y directly.) Prove that the $\hat{f}$-cost along any path in the search tree is nondecreasing.

Answers

a. (i) The search space has only 1 choice at each point: it is a long linear chain, but it is very deep. Depth first will do $O(n)$ work while iterative deepening will do $O(n^2)$.

(ii) The search space has a high branching factor, but there are goal states only 2 moves away from the start state, but for some other moves (including the first one tried) the search state is extremely deep or infinite.

(iii) All paths in the search space are finite, there are no goal states near the start state, but there is a goal state at the end of all branches. (Or: referring to a high branching factor causing BFS to exceed available memory.)

b. The triangle inequality applied to a heuristic $\hat{h}$ says that $\hat{h}(n) \leq h'(n, n') + \hat{h}(n')$ for any nodes n, n' (since the triangle inequality is true of any goal node g). Nondecreasing $\hat{f}$-cost along a path means that the f-cost of a successor node is always at least as large as that of the node itself.

Want: $\hat{f}(n) \leq \hat{f}(n')$ if $n' \in S(n)$, the successors of node n.

I.e., want (rewriting this in terms of g and $\hat{h}$): $g(n) + \hat{h}(n) \leq g(n') + \hat{h}(n')$ if $n' \in S(n)$.

Our aim is to show that this is implied by the triangle inequality. To do this, we simply add $g(n)$ to both sides of the triangle inequality:

$g(n) + \hat{h}(n) \leq g(n) + h'(n, n') + \hat{h}(n')$

But if n' is a successor of n, then $g(n) + h'(n, n') \leq g(n')$ (as h' is optimistic). Hence: $g(n) + \hat{h}(n) \leq g(n') + \hat{h}(n')$, as required.

2 Logic: resolution (12 points)

(This question comes from (the late) Jon Barwise and (our new provost) John Etchemendy. It also appears in the exercises of Russell and Norvig.)

Consider the following statements:

> If a unicorn is mythical, then it is immortal, but if it is not mythical, then it is a mortal mammal. If the unicorn is either immortal or a mammal, then it is horned. The unicorn is magical, if it is horned.

From the above, can you prove that the unicorn is mythical? How about magical? Horned? Use resolution for your proofs (using propositional logic is possible and acceptable). Show:

a. (2 points) the basic logical translation of this text

b. (2 points) the translations of these into a form suitable for resolution theorem proving

c. (8 points) For each of the three unicorn properties (mythical, magical, horned), if it can be proved, show your proof. If it can't be proved, explain the justification for being able to conclude that the fact that unicorns have this property doesn't follow from the information given.

Answers

a. All statements about unicorns, so we don't mention them in proposition names.

   (i) Mythical $\rightarrow$ Immortal (i.e., Mythical $\rightarrow \neg$ Mortal)

   (ii) $\neg$ Mythical $\rightarrow$ Mortal $\wedge$ Mammal (which is equivalent to $\neg$ Mythical $\rightarrow$ Mortal, $\neg$ Mythical $\rightarrow$ Mammal)

   (iii) (Immortal $\vee$ Mammal) $\rightarrow$ Horned (which is equivalent to (Mortal $\vee$ Horned) $\wedge$ ($\neg$ Mammal v Horned))

   (iv) Horned $\rightarrow$ Magical

b. The following clauses are the result (with obvious abbreviations):

   (i) $\neg$My $\vee$ $\neg$Mo

   (ii) My $\vee$ Mo

   (iii) My $\vee$ Mam

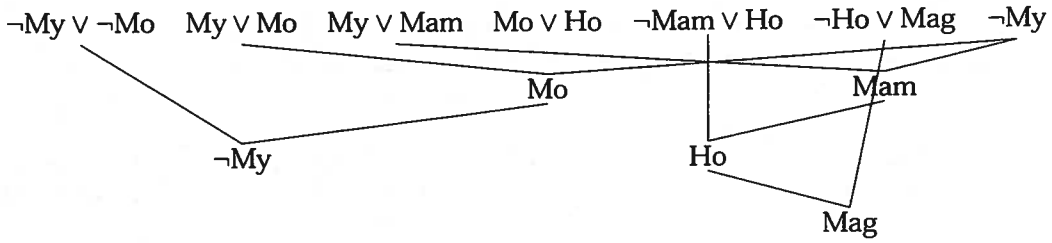
   (iv) Mo $\vee$ Ho

(v) ¬Mam ∨ Ho

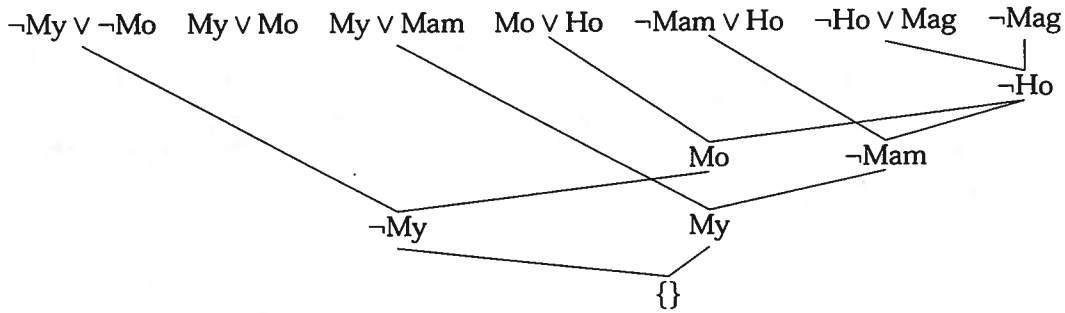
(vi) ¬Ho ∨ Mag

c. We use refutation proofs which will derive a contradiction if a set of sentences is unsatisfiable.

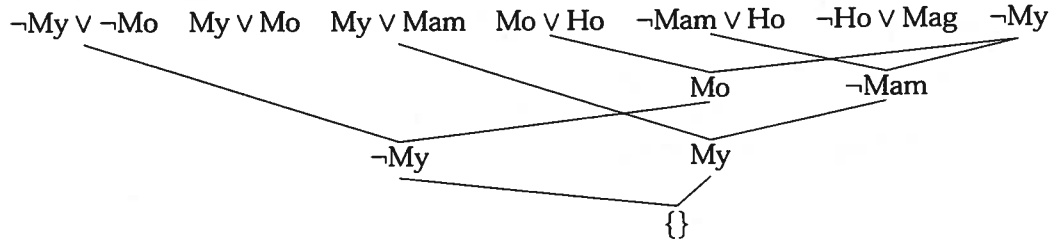
(i) Is it mythical? We add ¬My. No empty clause (i.e., no contradiction) can be derived. Refutation failure is a complete proof procedure for the propositional calculus, i.e., $KB \wedge \neg P \rightarrow False) \leftrightarrow (KB \rightarrow P)$. Hence, the given clauses do not entail that the unicorn is mythical.



(ii) Is it magical? We add ¬Mag. The empty clause (i.e., a contradiction) can be derived. Hence, the given clauses do entail that the unicorn is magical.



(iii) Is it horned? We add ¬Ho. The empty clause (i.e., a contradiction) can be derived. Hence, the given clauses do entail that the unicorn is horned.



3 Probabilistic models (12 points)

a. (1 point) A scientist tells us that 1/3 of all kangaroos have blue eyes, and 1/3 of all kangaroos are left-handed. On the basis of this information alone, what bounds can we place on the proportion of kangaroos that are both blue-eyed and left-handed?

b. (2 points) A loglinear model for the joint distribution of some number of categorical variables $X_1, \ldots, X_n$ takes the following general form:

$$\log P(X_1 = x_1, \ldots, X_n = x_n) = \sum_{C \in \mathcal{C}} \lambda_C(x_C)$$

where each $C \subseteq \{1, \ldots, n\}$ (so $\mathcal{C} \subseteq 2^{\{1,\ldots,n\}}$). That is, x_C is some subset of the x_i variables, and $\lambda_C(\cdot)$ is a function of this subset.

Show that all Bayesian networks ("graphical models") are loglinear models.

c. (9 points) Consider this situation: You saw John talking to his boss. Later you saw John looking upset. John may have been upset because his boss gave him a warning. Or he may be upset because his boyfriend left him.

(i) (1 point) Draw a suitable Bayesian network to represent the causal structure among the 4 random variables B for boss talking to John, W for boss warning John, U for John being upset, and L for his boyfriend leaving him.

(ii) (1 point) Make up conditional probability tables for each node. [HINT: keep the numbers simple. Use quarters, thirds, and halves!!]

(iii) (4 points) Given the information above (B and U), calculate the probability that John's boss warned him.

(iv) (3 points) Calculate the probability that John's boss warned him given the above information and that you know that his boyfriend left him.

Your answers may be approximate, but you should show your work

Answers

a. $0 \leq p \leq 1/3$

b. Suppose the Bayesian network has k nodes. In a Bayes net, due to the Markov assumption, the joint probability can be expressed as follows:

$$P(X_1 = x_1, \ldots, X_k = x_k) = \prod_{i=1}^{k} P(X_i | Pa(X_i))$$

where $Pa(X_i)$ are the parent nodes of X_i in the directed graph. It was accepted to say that the λ_C were these conditional probability functions, and the result is thus in the form required. But this was a little imprecise, since x_C was specified as a set, whereas doing things this way requires that one member of the set be distinguished.

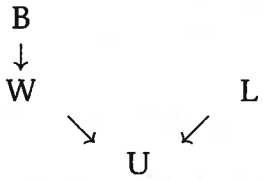
So, for $i = 1, \ldots, k$, let $C_i = \{X_i\} \cup Pa(X_i)$ and let $C_{k+i} = Pa(X_i)$. For $i = 1, \ldots, k$, let λ_{C_i} be the joint probability of the variables given as arguments, and for $i = k+1, \ldots, 2k$, let λ_{C_i} be the inverse of the joint probability of the variables given as arguments.

Then:

$$P(X_1 = x_1, \ldots, X_k = x_k) = \prod_{i=1}^{2k} \lambda_{C_i}(X_{C_i})$$

Taking logs of both sides we now have a loglinear model in the form required.

c. (i)



(ii) A possible answer

	B	~B
	0.1	0.9

	L	~L
	0.2	0.8

	W	~W
B	0.3	0.7
~B	0.1	0.9

	U	~U
W, L	0.9	0.1
W, ~L	0.8	0.2
~W, L	0.7	0.3
~W, ~L	0.1	0.9

(iii) We are interested in $P(W|B,U) = P(W,B,U)/P(B,U)$. From first principles:

$$P(B,L,W,U) = P(B)P(L)P(W|B)P(U|W,L) = 0.1 \times 0.2 \times 0.3 \times 0.9 = 0.0054$$
$$P(B,\neg L,W,U) = 0.1 \times 0.8 \times 0.3 \times 0.8 = 0.0192$$
$$P(B,L,\neg W,U) = 0.1 \times 0.2 \times 0.7 \times 0.7 = 0.0098$$
$$P(B,\neg L,\neg W,U) = 0.1 \times 0.8 \times 0.7 \times 0.1 = 0.0056$$

So,

$$P(W|B,U) = \frac{0.0054 + 0.0192}{0.0054 + 0.0192 + 0.0098 + 0.0056} = \frac{0.0246}{0.04} = 0.615$$

(iv) And

$$P(W|B,U,L) = \frac{0.0054}{0.0054 + 0.0098} = \frac{0.0054}{0.0152} \approx 0.355$$

The lower probability estimate here shows the phenomenon of "explaining away".

4 Learning (14 points)

a. (2 points) A circuit has two input values A and B whose values are either +1 for 'true' or -1 for 'false'. Design a perceptron (Linear Threshold Unit) network which computes the function not (A or B). Draw the network and indicate clearly all weights and threshold values (assume the network outputs 1 if the dotproduct of the weights and the inputs is greater than some threshold t specified by each unit).

b. (3 points) We are trying to predict whether Comps questions are easy or difficult ($D = +$ if they are difficult) based on two features:

L Whether they are long (1 = long)

M Whether they have a lot of math (1 = yes)

For training data, we have examined 12 Comps questions, and have collected the following statistics, which we show twice: on the left are counts for the diferent data patterns, and on the right the data is shown in a contingency table showing $-:+$ counts for each combination of the classificatory variables.

L	M	D	count
0	0	−	4
0	0	+	1
0	1	−	0
0	1	+	3
1	0	−	1
1	0	+	2
1	1	−	1
1	1	+	0

		M 0	M 1
L	0	4:1	0:3
	1	1:2	1:0

Draw a decision tree for this data (using information gain for node splitting, and no stopping criterion or pruning, so that the tree is grown so long as there is some classificatory feature that appears to have information about the target feature).

c. (4 points) A Naive Bayes classifier for this problem predicts the target feature from the prior and independently from the classificatory features as follows:

$$\text{Choose } \hat{d} = \arg\max_{d=\{-,+\}} P(d)P(L = l|d)P(M = m|d)$$

(That is, it calculates the expression shown for both $d = +$ and $d = -$ and chooses the value of d that gives the expression higher probability.) What classificatory decisions would a Naive Bayes classifier make for each combination of classificatory variables?

d. (3 points) Suppose we have 3 test data instances, whose correct classification we know, as follows:

L	M	D
0	0	−
1	1	+
0	1	+

What is the decision of each classifier on each datum? Which does better overall?

e. (2 points) Is the better performance of one learner reasonable or surprising here?

7 8

Adj → beautiful
N → city

(iii) *Sydney* sydney
is $\lambda P \lambda x.P(x)$
a
beautiful $\lambda x.$beautiful(x)
city $\lambda x.$city(x)

(iv) S(rel(subj)) → NP(subj) VP(rel)
NP(val) → PN(val) |
NP($\lambda x.$(adj$(x) \wedge$ n(x))) → (Det) Adj(adj) N(n)
VP(rel(obj)) → V(rel) NP(obj)

(v) This grammar just ignores the article *a*, but it would need a (probably different) semantic translation when *a* is used in a subject or object NP. The complement NP is here a property, and not quantificational as in most uses of NPs in natural languages. (Also, *is* is just an identity function, but is treated as a higher order function to make parallel to other verbs. Not all adjectives are intersective.)

ALTO

Comprehensive Exam: **Autumn 2000-01**

Automata and Formal Languages (60 points)

Problem 1. [10 points]
Consider the language L defined by the regular expression 00^*10. Provide a PDA M for this language using *as few states as possible*. Note that there is a PDA with only 1 state and that the number of points you get will depend on the number of states used in your solution.

Problem 2. [18 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.*

1. If L_1 and L_2 are both non-regular, then $L_1 \cap L_2$ must be non-regular.
2. $L = \{w \in \{a, b, c\}^* \mid w$ does not contain an equal number of occurrences of a, b, and $c\}$ is context-free.
3. Let L represent the language of a non-deterministic finite-state automaton N; then, swapping the final and non-final states of N gives a machine N' whose language is the complement of L.
4. Assume that P $\neq$ NP. If L_1 is in P and L_2 is in NP, then $L_1 \cap L_2$ must be in P.
5. If L_1 and L_2 are both in NP, then $L_1.L_2$ must be in NP.
6. If L_1 is context-free and L_2 is NP-complete, then $L_1 \cup L_2$ must be NP-complete.

Problem 3. [12 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*. You must give the *smallest* class that contains *every possible language* fitting the following definitions. For example, the language of a DFA M could be *empty* or *finite*, and must always be *context-free*, but the smallest class that is appropriate is *regular*.

1. The language $L = \{a^i b^j c^k d^l \mid i = k$ and $j = l\}$.
2. The set of strings from $\{0, 1\}^*$ which, when viewed as integers written in binary, are divisible by 3.
3. The language of a non-deterministic finite state automaton (NFA) with only two states.
4. The language of a non-deterministic push-down automaton (NPDA) with only one

5. The complement of a language L that belongs to P (polynomial time) but is not context-free.

6. A language L to which we can give a polynomial-time reduction from an undecidable language.

Problem 4. [10 points]
Using a reduction from a known undecidable problem, prove that the following problem is undecidable: Determine whether a Turing machine M halts on all inputs from $\{0,1\}^*$ that represent a valid encoding of some Turing machine. (You may assume any standard scheme for encoding a Turing machine into a string of 0's and 1's.)

Problem 5. [10 points]
Recall the decision problems called 2-SAT and 3-SAT. These are the versions of the satisfiability problems for 2-CNF and 3-CNF boolean formulas, respectively.

a). Prove that 2-SAT is polynomial-time reducible to 3-SAT. (Describe a reduction and justify its correctness.)

b). Given that 3-SAT is NP-complete, is the result in part (a) sufficient to prove the NP-completeness of 2-SAT? Explain.

AUTO
SOLUTIONS

Comprehensive Exam: **Autumn 2000-01**

Automata and Formal Languages (60 points)
Sample Solutions

Problem 1. [10 points]
Consider the language L defined by the regular expression 00*10. Provide a PDA M for this language using *as few states as possible.* Note that there is a PDA with only 1 state and that the number of points you get will depend on the number of states used in your solution.

Solution:
The following PDA with only 1 state accepts the language $L(00^*10)$ by empty stack. The PDA has the following components: $Q = \{q\}$, $\Sigma = \{0,1\}$, $\Gamma = \{Z_0, X, Y\}$, $q_0 = q$, and $F = \{\}$. The transition function is as follows:

$$\begin{aligned}
\delta(q,0,Z_0) &= \{(q,X)\} \\
\delta(q,0,X) &= \{(q,X)\} \\
\delta(q,1,X) &= \{(q,Y)\} \\
\delta(q,0,Y) &= \{(q,\epsilon)\}
\end{aligned}$$

Problem 2. [18 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.*

1. If L_1 and L_2 are both non-regular, then $L_1 \cap L_2$ must be non-regular.
2. $L = \{w \in \{a,b,c\}^* \mid w$ does not contain an equal number of occurrences of a, b, and $c\}$ is context-free.
3. Let L represent the language of a non-deterministic finite-state automaton N; then, swapping the final and non-final states of N gives a machine N' whose language is the complement of L.
4. Assume that P $\neq$ NP. If L_1 is in P and L_2 is in NP, then $L_1 \cap L_2$ must be in P.
5. If L_1 and L_2 are both in NP, then $L_1.L_2$ must be in NP.
6. If L_1 is context-free and L_2 is NP-complete, then $L_1 \cup L_2$ must be NP-complete.

Solution:

1. False
2. True

3. False

4. False

5. True

6. False

Problem 3. [12 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*. You must give the *smallest* class that contains *every possible language* fitting the following definitions. For example, the language of a DFA M could be *empty* or *finite*, and must always be *context-free*, but the smallest class that is appropriate is *regular*.

1. The language $L = \{a^i b^j c^k d^l \mid i = k \text{ and } j = l\}$.

2. The set of strings from $\{0,1\}^*$ which, when viewed as integers written in binary, are divisible by 3.

3. The language of a non-deterministic finite state automaton (NFA) with only two states.

4. The language of a non-deterministic push-down automaton (NPDA) with only one state.

5. The complement of a language L that belongs to P (polynomial time) but is not context-free.

6. A language L to which we can give a polynomial-time reduction from an undecidable language.

Solution:

1. Recursive

2. Regular

3. Regular

4. Context-free

5. Recursive

6. Recursively-enumerable

Problem 4. [10 points]
Using a reduction from a known undecidable problem, prove that the following problem is undecidable: Determine whether a Turing machine M halts on all inputs from $\{0,1\}^*$ that represent a valid encoding of some Turing machine. (You may assume any standard scheme for encoding a Turing machine into a string of 0's and 1's.)

Solution:

Let $L_M = \{< M > |\ M$ halts on any input w which is a valid encoding of a Turing machine$\}$.

The halting problem is undecidable: Determine whether a Turing machine M halts on a particular input w. We define the language $L_H = \{< M, w > |\ M$ halts on $w\}$ to represent the halting problem.

We will give a reduction from L_H to L_M, thereby establishing the undecidability of L_M. Given an instance of the halting problem, say M and w, the reduction constructs a new Turing machine M'. The machine M' ignores its input and simulates M on input w. Clearly, if M halts on w, then M' halts on all w' which are valid encodings of Turing machines (in fact, it halts on all inputs w'). Furthermore, if M does not halt on w, then M' does not halt on any w'. It follows that $< M, w > \in L_H$ if and only if $M' \in L_M$, establishing the validity of the reduction. Since the reduction is easy to compute, it follows that L_M is undecidable.

Problem 5. [10 points]
Recall the decision problems called 2-SAT and 3-SAT. These are the versions of the satisfiability problems for 2-CNF and 3-CNF boolean formulas, respectively.

a). Prove that 2-SAT is polynomial-time reducible to 3-SAT. (Describe a reduction and justify its correctness.)

b). Given that 3-SAT is NP-complete, is the result in part (a) sufficient to prove the NP-completeness of 2-SAT? Explain.

Solution:

a). We describe a polynomial-time reduction from 2-SAT to 3-SAT. Given a 2-SAT formula $F(X_1, \ldots, X_n)$, the reductions a 3-SAT formula $G(X_1, \ldots, X_n, Z, A, B)$ as follows.

We create 3 new variables Z, A, and B. For each clause $X_i \cup X_j$ in F, we create a clause $X_i \cup X_j \cup \overline{Z}$ in G. Also, we add to G four additional clauses: $Z \cup A \cup B$, $Z \cup \overline{A} \cup B$, $Z \cup A \cup \overline{B}$, and $Z \cup \overline{A} \cup \overline{B}$. Quite clearly, the reduction can be computed in polynomial time. We now establish the validity of the reduction by showing that F has a satisfying truth assignment if and only if G has a satisfying truth assignment.

If F has a satisfying truth assignment, we can get a satisfying truth assignment for G as follows: use the same truth values for $X_1, \ldots, X_n$ and Z, A, B to TRUE don't care about A and B). It is easy to verify that G is satisified by this truth assignment.

If G has a satisfying truth assignment, then Z must be set to TRUE; otherwise, there is no way to satisfy the four additional clauses. It follows that the same truth assignment, restricted to $X_1, \ldots, X_n$, is a satisfying truth assignment for F.

b). No, this is not sufficient to prove the NP-completeness of 2-SAT. A reduction from 3-SAT to 2-SAT would have implied the NP-hardness of 2-SAT, but this reduction is in the reverse direction. In fact, 2-SAT can be solved in polynomial time and hence is unlikely to be NP-complete.

COMP

Compilers Comprehensive, November 2, 2000

This is a 60 minute open book exam (but do not use computers). All answers should be written in the blue book (not on the exam). Clear, organized answers to essay questions is important.

1. (10 points) In your blue book, indicate whether each language is LL(1), LR(1), both, or neither.

	LL(1)?	LR(1)
(a) $S \rightarrow aSa \mid aSb \mid c$	no. first(aSa) ∩ first(aSb) ≠ ∅	yes
(b) $S \rightarrow Sa \mid bS \mid c$	no. left recursive	no
(c) $S \rightarrow aSa \mid bS \mid c$	yes, no problems by process of elim	yes
(d) $S \rightarrow aSa \mid bSb \mid \epsilon$	no, first(aSa) ∩ Follow(S) ≠ ∅ and S→ε	no
(e) $S \rightarrow Sa \mid b \mid c$	no, left recursive	yes

2. (20 points)

The following YACC grammar parses simple Polish expressions (e.g., `*+12+34`, which is equal to 21).

Show what actions need to be added to print the equivalent reverse Polish expressions (e.g., `12+34+*`). Assume that the lexical analyzer returns the numerical value of the number described by `NUM`

```
%token NUM

%%

S       :       R
        ;

R       :       NUM  { printf("%d ", $1); }
        |       '+' R R
        |       '*' R R
        ;
```

Now add actions to print Polish expressions, given reverse Polish:

```
S       :       R
        ;

R       :       NUM
        |       R R '+'
        |       R R '*'
        ;
```


34

2000

3. (20 points) Describe some well-known compiler optimizations that could be usefully applied to the following code, and why they would improve the code. Be specific about what parts of the code would be improved and why. The most basic optimizations will count most heavily in grading this question.

```
struct s {
  int f1;
  double f2;
};

struct s A[100];

extern f(struct s *);

int main(void)
{
  int i;
  for (i=1; i<100; i++) {
    A[i].f1 = A[i-1].f1 - 1;
  }

  return f(A);
}
```

4. (10 points) This question asks for practical reasons to prefer one way of writing parser over another.

   (a) Give three good reasons to write a recursive descent parser by hand, even though highly efficient automatic parser generators are freely available.

   (b) Give three good reasons to use an LALR parser generator such as YACC instead of writing a recursive descent parser by hand.

35

COMP SOLUTIONS.

Solutions to Compilers Comprehensive, October 2000

This is a 60 minute open book exam (but do not use computers). All answers should be written in the blue book (not on the exam). Clear, organized answers to essay questions is important.

1. (10 points) In your blue book, indicate whether each language is LL(1), LR(1), both, or neither.

   (a) $S \rightarrow aSa|aSb|c$

   (b) $S \rightarrow Sa|bS|c$

   (c) $S \rightarrow aSa|bS|c$

   (d) $S \rightarrow aSa|bSb|\epsilon$

   (e) $S \rightarrow Sa|b|c$

   Solution

   There were two interpretations of this problem: (1) Can the grammar be rewritten into an equivalent form that has the desired property; (2) does the grammar have the desired property as it is given. I intended the second interpretation, but graded by guessing what interpretation the student had in mind.

   Here are the answers for the first interpretation:

   (a) $S \rightarrow aSa|aSb|c$ *both*

   (b) $S \rightarrow Sa|bS|c$ *both*

   (c) $S \rightarrow aSa|bS|c$ *both*

   (d) $S \rightarrow aSa|bSb|\epsilon$ *neither*

   (e) $S \rightarrow Sa|b|c$ *both*

   Here are the answers for the second intepretation:

   (a) $S \rightarrow aSa|aSb|c$ *LR(1) (not LL(1) because not left factored)*

   (b) $S \rightarrow Sa|bS|c$ *Neither (ambiguous)*

   (c) $S \rightarrow aSa|bS|c$ *LL(1) and LR(1)*

   (d) $S \rightarrow aSa|bSb|\epsilon$ *Neither (not a deterministic CFL)*

   (e) $S \rightarrow Sa|b|c$ *LR(1) but not LL(1) (left recursion)*

3.

2. (20 points)

The following YACC grammar parses simple Polish expressions (e.g., `*+12+34`, which is equal to 21).

Show what actions need to be added to print the equivalent reverse Polish expressions (e.g., `12+34+*`). Assume that the lexical analyzer returns the numerical value of the number described by `NUM`

```
%token NUM

%%

S       :       R
        ;

R       :       NUM  { printf("%d ", $1); }
        |       '+' R R
        |       '*' R R
        ;
```

Solution:

```
S       :       R
        ;

R       :       NUM { printf("%d ", $1); }
        |       '+' R R { printf("+ "); }
        |       '*' R R { printf("* "); }
        ;
```

Now add actions to print Polish expressions, given reverse Polish:

```
S       :       R
        ;

R       :       NUM
        |       R R '+'
        |       R R '*'
        ;
```


37

Solution.

If we use the same actions from the previous solution, the result prints in RPN, not PN (the operators are printed after the reductions of the R's).

Here's something else that doesn't work:

```
...
|       { print("+ "); } R R '+'
```

The problem is a shift/reduce conflict: shift *NUM* (from $R \rightarrow NUM$) or reduce the implicit production $M_1 \rightarrow \epsilon$ that was added for the new action?

It is difficult to output the result on-the-fly during parsing because the last operator in the input has to be printed first. We have to parse the whole input before we can print anything. The solution before builds up the entire output in a string, then prints it at the end of parsing. Another approach would be to build a tree and then print it in PN.

```
S       :       R { printf("%s\n", $1); }
        ;

R       :       NUM
                {
                  char *str = (char *) calloc(1,12);
                  sprintf(str, "%d", $1); $$ = str;
                }
        |       R R '+'
                {
                  char *str = (char *) calloc(1, strlen($1) + strlen($2) + 1);
                  sprintf(str, "+ %s %s", $1, $2); $$ = str;
                }
        |       R R '*'
                {
                  char *str = (char *) calloc(1, strlen($1) + strlen($2) + 1);
                  sprintf(str, "* %s %s", $1, $2); $$ = str;
                }
        ;
```


38

3. (20 points) Describe some well-known compiler optimizations that could be usefully applied to the following code, and why they would improve the code. Be specific about what parts of the code would be improved and why. The most basic optimizations will count most heavily in grading this question.

```
struct s {
  int f1;
  double f2;
};

struct s A[100];

extern f(struct s *);

int main(void)
{
  int i;
  for (i=1; i<100; i++) {
    A[i].f1 = A[i-1].f1 - 1;
  }

  return f(A);
}
```

Solution

There is significant variation in terminology, and sometimes the same effect can be achieved from different avenues. Here is a sample answer.

Unoptimized, computing the address of `A[i]` requires generating code to do something like: `globalsaddress + Aoffset + 12 * 4 * fetch(stackpointer + localsoffset + ioffset)` where "globalsadress", "Aoff", "localsoffset" and "ioffset" are all constant. *Constant-folding* could compute `globalsaddress+Aoffset`, `localsoffset+ioffset`, and `12*4` before the program is run (at compile time and load time).

`stackpointer+localsoffset+ioffset` (getting the variable i) will appear twice, so *Common subexpression elimination* could be used to compute it only once (and save the result for later re-use). A sophisticated compiler could attempt to do algebraic transformations to make the address of A[i].f1 into a common subexpression (A[i-1].f1 is a constant offset from this), but re-arranging expressions in the right way is not necessarily easy.

The loop optimization *Reduction in strength* could be used to avoid multiplying by 48 every time the array is indexed in the loop (instead, 48 could be added to the array address each time).

If the loop were unrolled, there would be an opportunity to eliminate a different common subexpression, since the address of A[i] on one iteration is the same as A[i-1] on the next.

4. (10 points) This question asks for practical reasons to prefer one way of writing parser over another.

(a) Give three good reasons to write a recursive descent parser by hand, even though highly efficient automatic parser generators are freely available.

Solution:

- The grammar is simple, and I don't want to require people building the system to have YACC.
- No good parser generator is available (e.g., in the early days of Java).
- Greater flexibility, e.g., to look ahead more than one symbol or use other information to decide on an action.
- Error recovery may be more flexible and understandable.
- etc.

(b) Give three good reasons to use an LALR parser generator such as YACC instead of writing a recursive descent parser by hand.

Solution:

- It's easier, because parser construction is automatic.
- LALR(1) parsing is more powerful than recursive descent, which is basically LL(1). E.g., left recursion in the grammar works.
- The context free grammar is more readable, and is easier to make consistent with the source language specification.
- The resulting parser is likely to be faster.
- etc.

40

Computer Science Comprehensive Examination
Computer Architecture
[60 points]

This examination is open book. Please do all of your work on these sheets. Do not do your work in a blue book.

Number: _____

Problem	Max Score	Your Score
1	30	
2	24	
3	24	
4	22	
TOTAL	100	

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)

Problem 1 : Short Answer [3 points each, 30 points total]

A. Compared to an 8K direct-mapped cache, what type of misses will a 16K direct-mapped cache have fewer of? Circle all that apply.

(a) compulsory
(b) conflict
(c) capacity

B. Which instruction set is better able to express instruction-level parallelism?

(a) An accumulator instruction set
(b) A three-address general-register instruction set

C. Which mean should be used to combine execution speeds expressed in instructions/sec? Circle all that apply.

(a) Arithmetic mean
(b) Geometric mean
(c) Harmonic mean

D. A machine with register renaming is able to reorder instructions without regard to what type of dependencies? Circle all that apply.

(a) Data dependencies RAW
(b) Output dependencies WAW
(c) Anti-dependencies WAR
(d) Control dependencies branch

E. Adding accurate branch prediction to a processor reduces the impact on performance of which pipeline latency? Circle all that apply. Assume that branch target and branch condition are computed during the execution stage of the pipeline.

(a) From the completion of execution to where the results are available from a register.
(b) From the register stage to where the results of execution are available
(c) From the register stage to where the results of a memory load are available
(d) From the fetch stage to the register stage.

F. In the steady state, what will be the prediction accuracy of a one-bit branch predictor on the repeating sequence TTTTNTTTN (T=taken, N=not taken)?

5/9 = 56%

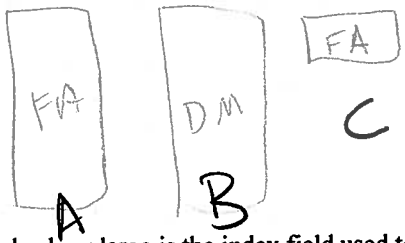
F D R E M W

G. An instruction for protected subsystem entry (sometimes called system call) must change which subset of the following five things atomically? (circle all that apply)

(a) The contents of a data register
(b) The program counter (sometimes called instruction pointer)
(c) The contents of the page table
(d) The contents of the cache
(e) The privilege level

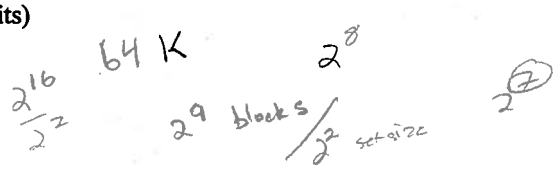
H. Suppose that cache A, a 16K-byte fully-associative cache, cache B, a 16K-byte direct-mapped cache, and cache C, a 4K-byte fully-associative cache are all referenced with an identical address sequence. All caches use a true least-recently used (LRU) replacement policy. Which of the following statements are true: (circle all that apply)

X(a) A will contain a superset of the data in B
(b) A will contain a superset of the data in C
X(c) B will contain a superset of the data in A
X(d) B will contain a superset of the data in C



I. In a 64K-byte four-way set-associative cache with 128-byte blocks, how large is the index field used to address the cache array? (write down the number of bits)

7



J. If the cache of question I is physically tagged and physical addresses are 40-bits long, what is the minimum possible length the tag may be for correct operation? (write down the number of bits)

26

4/3

Problem 2 : Pipeline Architecture [24 points total]

Consider a memory-to-memory machine with the following pipeline stages:

IF	Instruciton Fetch	Fetch instructions
SA	Source Address	Compute address of source operands
DF	Data Fetch	Fetch source operands from memory and/or registers
X	Execute	Execute arithmetic operations and compute destination address
DS	Data Store	Store results to memory or register

The destination operand for each instruction may be either a memory location or a register. Also, at most one source operand may be a memory location. The second source operand is always a register. Thus, for each operation type, op, this machine provides the following four types of instructions:

```
RR    R <- R op R        Register to register
MR    R <- R op M        Memory to register
RM    M <- R op R        Register to memory
MM    M <- R op M        Memory to memory
```

A. (8 Points) Assume for now that each pipeline stage operates in one clock cycle, that there are no cache misses, that there is bypassing of registers but not of memory, that there is no memory disambiguation, that a store must complete before a load to the same location can take place, and that there are no resource conflicts. Consider the following instruction sequence:

```
1.    8(R3) <- R4 + R5
2.    R6    <- R2 + 16(R7)
3.    12(R1) <- R6 + 8(R3)
4.    4(R1) <- R6 + 4(R3)
```

Note that because there is no disambiguation, the hardware cannot tell whether 8(R3) (the address formed by adding 8 to the contents of R3) is the same as 16(R7). Draw a pipeline diagram showing this code executing and show all data dependencies. (Hint: Your pipeline diagram should have a row for each instruction and a column for each clock cycle – you may use the table below as a guide or draw freehand. Indicate each dependency with an arrow between the source and destination).

12

1	IF	SA	DF	X	DS											
2		IF	SA	—	—	DF	X	DS								
3			IF	—	—	SA	DF	X	DS							
4				—	—	IF	SA	—	—	DF	X	DS				

B. (8 Points) Now assume that the hardware does do disambiguation and as soon as the addresses are calculated is able to show that 8(R3) is a different address than 16(R7). Draw the pipeline diagram for this case, again showing dependencies. (If you need to you may also assume that 12(R1), 4(R1), and 4(R3) are also different addresses).

IF	SA	DF	X	DS											
	IF	SA	DF	X	DS										
		IF	SA	—	DF	X	DS								
			IF	—	SA	DF	X	DS							

C. (8 Points) Now, starting with the machine of part B, assume that there is only a single memory port that must be shared between the DF and DS stages. Again draw the pipeline diagram for this case showing dependencies and identifying resource conflicts.

IF	SA	DF	X	DS											
	IF	SA	DF	X	DS										
		IF	SA	—	—	DF	X	DS							
			IF	—	—	SA	DF	X	DS						

3. Virtual memory [24 points total]

Consider a hypothetical machine with one-byte virtual addresses consisting of a 4-bit page field and a 4-bit offset field. The page field of a virtual address references a page table stored in page 0 of physical memory. Each entry of the page table is either the index of the page frame in physical memory that contains the page in question or the constant FF if the page is not in physical memory. At a given point in time the page 0 of physical memory has the following entries (all numbers are in hexadecimal):

0: 07	8: 06
1: 01	9: 02
2: 03	A: FF
3: FF 06	B: 05
4: 00	C: FF
5: FF	D: FF
6: FF	E: FF
7: FF	F: 04

A. [4 points] What physical address, if any, corresponds to virtual address 17 (hex)?

17

B. [4 points] What virtual address, if any, corresponds to physical address 47 (hex)?

F7

C. [4 points] What physical address, if any, corresponds to virtual address 77 (hex)?

none

D. [4 points] Is the mapping from virtual addresses to physical addresses one-to-one? Explain your answer?

No many v.a. can map to same p.a.

E. [8 points] To what virtual address would one write to, and what value should be written to that location to map virtual addresses 30-3F to physical addresses 60-6F?

30-3F

60-6F

43

virtual

write "06" into 43

43

4/6

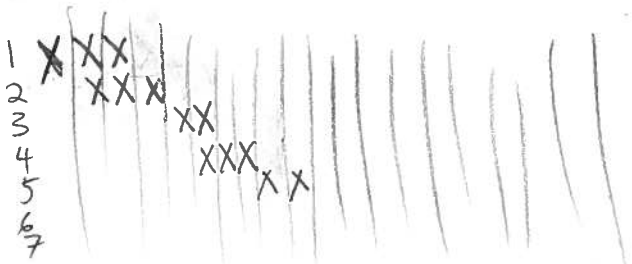
4. Instruction Issue [22 points total]

Consider the following instruction sequence:

```
1       LD      X,R1
2       LD      Y,R2
3       ADD     R1,R2,R3        ; R3 <- R1 + R2
4       LD      Z,R4
5       ADD     R3,R4,R5        ; R5 <- R3 + R4
6       MUL     R5,3,R6         ; R6 <- R5 * 3
7       ADD     R2,R4,R7        ; R7 <- R2 + R4
```

You may assume that all arithmetic operations have two-cycle latency and all loads have three-cycle latency. That is, the result of an arithmetic (memory) operation is available two (three) cycles after that operation enters the first execution stage of the machine. Also assume that there is full bypassing, that all loads hit in the cache, and that an arbitrary number of loads (and arithmetic operations) can be in flight at a single time. Hint: you need only consider the execution and memory stages of the pipeline to answer this question.

A. [6 points] How many cycles does this sequence take to execute on a single-issue in-order machine? Measure time from the cycle that the first instruction issues (enters the execution stage) to the cycle in which the result of the last instruction is available for use. Show your work.



B. [5 points] If issue order and resources are not limited, what is the shortest time this sequence of instructions could take? Show your work.

C. [5 points] How many cycles does this sequence take to execute on an in-order machine with multiple issue (assume an issue width as wide as you need)? Show your work

D. [6 points] Can you statically reorder the code to give the wide in-order machine of part C the performance bound of part B? If so, show the new ordering.

DB

Stanford University Computer Science Department
2000 Comprehensive Exam in Databases

- The exam is *open book and notes.*
- Answer all 6 questions on the exam paper itself, in the space provided for each question.
- The total number of points is 60; questions may have differing point values.
- You have 60 minutes to complete the exam (i.e., one minute per point). It is suggested you review the entire exam first, in order to plan your strategy.
- *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.

Provide your magic number here: _____

1	2	3	4
5	6		

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.

Problem 1: (10 points) Suppose relation $R(A, B, C, D, E)$ has functional dependencies

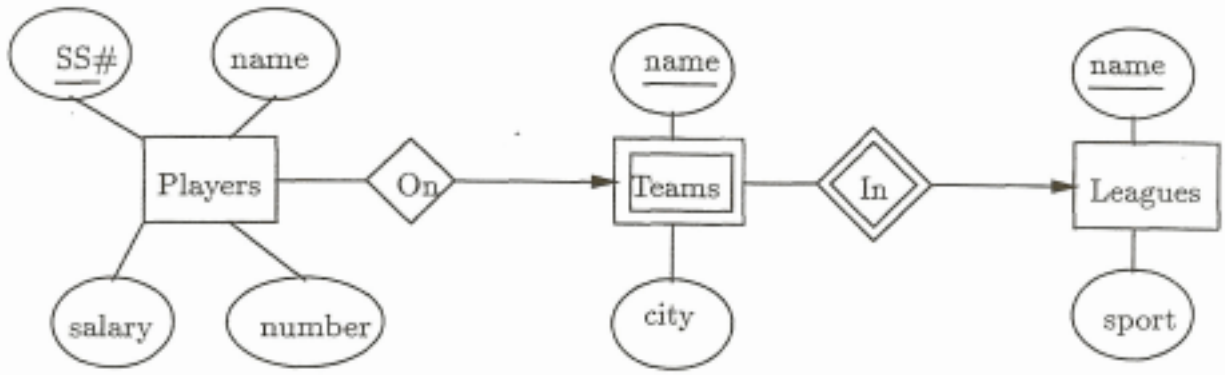
$A \rightarrow B$
$BC \rightarrow D$
$BE \rightarrow C$
$AD \rightarrow E$
$CE \rightarrow A$

a) What are all the keys of R?

b) Of the five given FD's, which violate the BCNF condition?

c) Of the five given FD's, which violate the 3NF condition?

Problem 2: (10 points) Below is an entity-relationship diagram:



a) Choose a relational database schema that represents this E/R diagram as faithfully as possible. Do not use a relation if its information is sure to be contained in some other relation of your schema.

b) Suppose that we delete the SS# attribute from *Players*. Exploit the fact that a team will not give the same number to two players in order to find a similar E/R diagram with a suitable key for *Players*. Draw your diagram below:

c) Convert your E/R diagram from (b) to an appropriate relational database schema.

Problem 3: (10 points) A database system, containing of two objects A and B executes three transactions: T_1, T_2 and T_3. Initially A has a value of 10, and B a value of 20.

T_1: First writes a value of 30 for A; then changes B to 40. Finally, T_1 commits. T_1 runs with isolation level `SERIALIZABLE`.

T_2: Starts by changing A to 50, then modifies B to 60. At this point T_2 does a rollback, and all changes are undone. T_2 runs with isolation level `SERIALIZABLE`.

T_3: Is a read-only transaction that first reads A and then reads B. The isolation level of T_3 is discussed below.

We do *not* know in what order these three transactions execute.

a) Assume that T_3 runs with isolation level `SERIALIZABLE`. What are all the possible A, B values that T_3 can read? Give each answer as a pair $[X, Y]$, where X is the A value read, and Y is the B value read by T_3.

---

b) Assume that T_3 runs with isolation level `READ COMMITTED`. What additional A, B values can T_3 read? [Do not list pairs given in part (a).]

---

c) Assume that T_3 runs with isolation level `READ UNCOMMITTED`. What additional A, B values can T_3 read? [Do not list pairs given in parts (a) or (b).]

Problem 4: (5 points) Suppose we have a relation with schema $ABCD$, the functional dependency $A \rightarrow B$ and the multivalued dependency $A \twoheadrightarrow C$. Give five other nontrivial multivalued dependencies that R must satisfy (i.e., MVD's that follow logically from the two given dependencies). Note that for a MVD to be nontrivial, its left and right sides must have no attribute in common, and there must be some attribute that is in neither the left nor right side.

Problem 5: (20 points) Consider the following relation:

```
Advised(Advisor, Student, Year)
```

A tuple (A, S, Y) in `Advised` specifies that advisor A advised student S who graduated in year Y. Assume that `Student` is a key for this relation.

a) Consider the following SQL query, which finds all advisors who advised a student who graduated in the same year that Hector Garcia-Molina or Jennifer Widom graduated.

```
SELECT Advisor
FROM Advised
WHERE Year IN
    (SELECT Year FROM Advised
    WHERE Student = 'Hector Garcia-Molina'
        OR Student = 'Jennifer Widom')
```

Write an SQL query, *without any subqueries* and *without the keyword* `DISTINCT`, that always produces the same set of tuples as the above query.

---

b) Are there are any circumstances in which your "equivalent" query can produce an answer different from that of the query above? Explain, if so.

c) Using SQL3 recursion, write a query that finds all "descendants" of Jeff Ullman, i.e., all students whose advisor was Jeff Ullman, or whose advisor's advisor was Jeff Ullman, or whose advisor's advisor's advisor was Jeff Ullman, and so on.

d) Write a SQL query that finds the advisor(s) with the longest advising span, i.e., with the longest period from their earliest advisee to their latest advisee.

Problem 6: (5 points) Two relations $A(x)$ and $B(y)$ each contain integers (i.e., their tuples have one component, which is an integer). Give `CREATE TABLE` statements for A and B with `CHECK` clauses sufficient to assure that $A \cap B$ will always be empty.

DB
SOLUTIONS

Stanford University Computer Science Department
2000 Comprehensive Exam in Databases

WITH ANSWERS

- The exam is *open book and notes.*
- Answer all 6 questions on the exam paper itself, in the space provided for each question.
- The total number of points is 60; questions may have differing point values.
- You have 60 minutes to complete the exam (i.e., one minute per point). It is suggested you review the entire exam first, in order to plan your strategy.
- *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.

Provide your magic number here: _____

1	2	3	4
5	6		

Problem 1: (10 points) Suppose relation $R(A, B, C, D, E)$ has functional dependencies

$A \rightarrow B$
$BC \rightarrow D$
$BE \rightarrow C$
$AD \rightarrow E$
$CE \rightarrow A$

a) What are all the keys of R?

Answer: AC, AD, AE, BE, and CE.

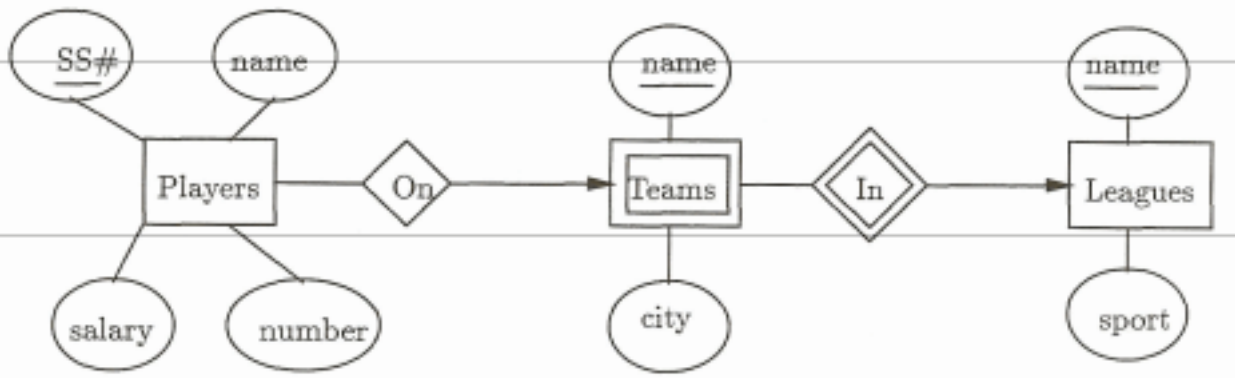
b) Of the five given FD's, which violate the BCNF condition?

Answer: The first two.

c) Of the five given FD's, which violate the 3NF condition?

Answer: None.

Problem 2: (10 points) Below is an entity-relationship diagram:

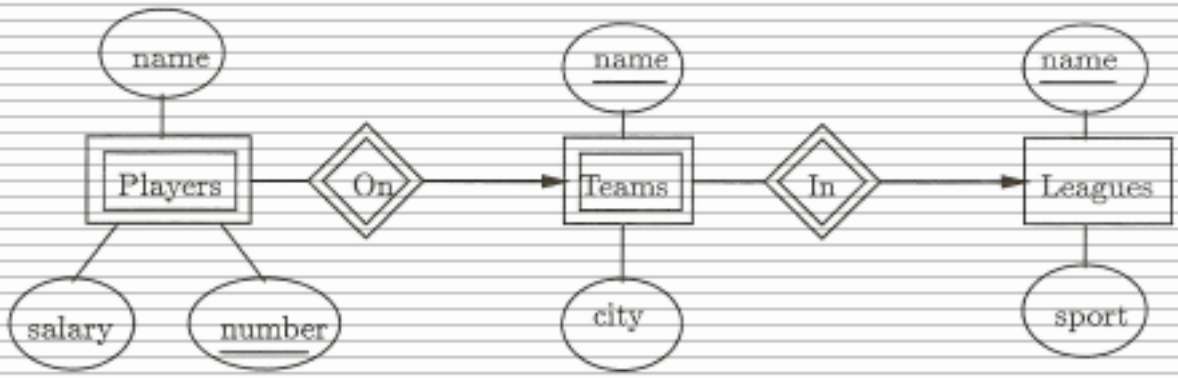


a) Choose a relational database schema that represents this E/R diagram as faithfully as possible. Do not use a relation if its information is sure to be contained in some other relation of your schema.

Answer:

```
Players(SS#, name, number, salary)
On(SS#, teamName, leagueName)
Teams(teamName, leagueName, city)
Leagues(name, sport)
```

b) Suppose that we delete the SS# attribute from *Players*. Exploit the fact that a team will not give the same number to two players in order to find a similar E/R diagram



c) Convert your E/R diagram from (b) to an appropriate relational database schema.

Answer:

```
Players(name, teamName, leagueName, number, salary)
Teams(teamName, leagueName, city)
Leagues(name, sport)
```

Problem 3: (10 points) A database system, containing of two objects A and B executes three transactions: T_1, T_2 and T_3. Initially A has a value of 10, and B a value of 20.

T_1: First writes a value of 30 for A; then changes B to 40. Finally, T_1 commits. T_1 runs with isolation level `SERIALIZABLE`.

T_2: Starts by changing A to 50, then modifies B to 60. At this point T_2 does a rollback, and all changes are undone. T_2 runs with isolation level `SERIALIZABLE`.

T_3: Is a read-only transaction that first reads A and then reads B. The isolation level of T_3 is discussed below.

We do *not* know in what order these three transactions execute.

a) Assume that T_3 runs with isolation level `SERIALIZABLE`. What are all the possible A, B values that T_3 can read? Give each answer as a pair $[X,Y]$, where X is the A value read, and Y is the B value read by T_3.

Answer: [10,20], [30,40].

b) Assume that T_3 runs with isolation level `READ COMMITTED`. What additional A, B values can T_3 read? [Do not list pairs given in part (a).]

Answer: [10,40].

c) Assume that T_3 runs with isolation level `READ UNCOMMITTED`. What additional A, B values can T_3 read? [Do not list pairs given in parts (a) or (b).]

Answer: [50,60], [50,20], [50,40], [30,20], [10,60], [30,60].

Problem 4: (5 points) Suppose we have a relation with schema $ABCD$, the functional dependency $A \rightarrow B$ and the multivalued dependency $A \twoheadrightarrow C$. Give five other nontrivial multivalued dependencies that R must satisfy (i.e., MVD's that follow logically from the two given dependencies). Note that for a MVD to be nontrivial, its left and right sides must have no attribute in common, and there must be some attribute that is in neither the left nor right side.

Answer: Among other choices are anything with a left side A and a right side that is any subset of $\{B, C, D\}$ except for $\{C\}$ (because it is given) and $\{B, C, D\}$ (because it is trivial).

Problem 5: (20 points) Consider the following relation:

```
Advised(Advisor, Student, Year)
```

A tuple (A, S, Y) in `Advised` specifies that advisor A advised student S who graduated in year Y. Assume that `Student` is a key for this relation.

a) Consider the following SQL query, which finds all advisors who advised a student who graduated in the same year that Hector Garcia-Molina or Jennifer Widom graduated.

```
SELECT Advisor
FROM Advised
WHERE Year IN
    (SELECT Year FROM Advised
    WHERE Student = 'Hector Garcia-Molina'
        OR Student = 'Jennifer Widom')
```

Write an SQL query, *without any subqueries* and *without the keyword* `DISTINCT`, that always produces the same set of tuples as the above query.

Answer:

```
SELECT A1.Advisor
FROM Advised A1, Advised A2
    WHERE A1.Year = A2.Year
        AND (A2.Student = 'Hector Garcia-Molina'
            OR A2.Student = 'Jennifer Widom');
```

b) Are there are any circumstances in which your "equivalent" query can produce an answer different from that of the query above? Explain, if so.

Answer: If Hector and Jennifer graduated in the same year Y, then this query will return two copies of an advisor for each student he graduated in year Y, while the original query will return only one copy.

c) Using SQL3 recursion, write a query that finds all "descendants" of Jeff Ullman, i.e., all students whose advisor was Jeff Ullman, or whose advisor's advisor was Jeff Ullman, or whose advisor's advisor's advisor was Jeff Ullman, and so on.

Answer:

```
WITH RECURSIVE Descendant AS (
    (SELECT Student FROM Advised WHERE Advisor = 'Jeff Ullman')
        UNION
    (SELECT A.Student FROM Advised A, Descendant D
    WHERE A.Advisor = D.Student)
);
SELECT * FROM Descendant;
```

d) Write a SQL query that finds the advisor(s) with the longest advising span, i.e., with the longest period from their earliest advisee to their latest advisee.

Answer:

```
SELECT Advisor
FROM Advised
GROUP BY Advisor
HAVING (Max(Year) - Min(Year)) >= ALL(
    SELECT Max(Year) - Min(Year)
    FROM Advised
    GROUP BY Advisor
);
```

Problem 6: (5 points) Two relations $A(x)$ and $B(y)$ each contain integers (i.e., their tuples have one component, which is an integer). Give `CREATE TABLE` statements for A and B with `CHECK` clauses sufficient to assure that $A \cap B$ will always be empty.

Answer:

```
CREATE TABLE A(
    x INT CHECK(x NOT IN (SELECT y FROM B)
);
CREATE TABLE B(
    y INT CHECK(y NOT IN (SELECT x FROM A)
);
```

Computer Graphics Comprehensive Exam

Computer Science Department
Stanford University
Fall 2000

NAME:

Note: This is exam is *open-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 this exam.

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.

_____

1. [Total: 25 points] Color.

1A. (5 points) Monitors and other display devices often use three color components, such as R, G and B. Lights, however, are described by their power spectra; that is the amount of power per unit wavelength interval. Why are only three color components sufficient to represent a given color?

1B. (5 points) Suppose two colored light sources are directed at the same spot on a perfectly reflective diffuse white wall. Derive the color components of the reflected light?

1C. (10 points) A colored filter such as a piece of cellophane may be characterized by its transmission spectrum. The transmission spectrum is defined to be the percentage of light (represented as a number between 0 and 1) passing through the filter per unit wavelength interval. Suppose two filters are stacked on top of each other and placed over an ideal white light with a flat spectrum (that is, the same amount of light energy is present at each wavelength). Derive the color of the transmitted light in terms of the spectra of the filters? How are the color components (RGB) of the transmitted light related to the color components of the filters?

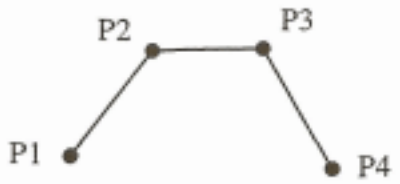
2. [Total: 20 points] Transformations.

2A. (10 points) Consider transforming a point in 2D. Derive a transformation matrix that corresponds to first rotating the point by 45 degree and then translating it by (1,1).

2B. (10 points) The above transformation may be executed in the reverse order. That is, by first translating and then rotating. Derive the amount of rotation and translation that must be applied to have the same effect as the sequence in 2A.

3. [Total: 20 points] Bezier curves.

3A. [10 points] Drawn below are four points P1, P2, P3, and P4 that can be used to define a cubic Bezier curve P(t). Describe an algorithm to evaluate point P(t) given t.



3B. (10 points) Suppose you wanted to find an intersection of a Bezier curve with a line. Describe a recursive algorithm that will find *all* the intersections of the curve with the line. What properties of the Bezier curve form the basis of your algorithm?

4. [Total: 20 points] Reflection models.

Consider reflection off a surface in 2D. By 2D we mean the surface is actually a curve segment all directions and points are in the plane (think flatland). There are several important directions: **L**, the direction to the light source; **E**, the direction to the eye; **H**, the half way vector between **L** and **E** (that is, the direction what would reflect **L** to **E**, and vice versa); and the normal **N**. Also, consider the reflection operator **R**(**D**,**N**) which reflects any direction **D** about the normal **N**. For the purposes of this problem, consider **L**, **E**, **H**, and **N** to have unit length.

4A. (5 points) Derive an expression for the reflection operator **R**(**D**,**N**) in terms of vector algebra.

4B. (15 points) There are two common reflection models used to model highlights, the Phong Model and the Blinn Model. The Phong Model uses $(\mathbf{R}(\mathbf{E},\mathbf{N}) \bullet \mathbf{L})^s$ and the Blinn Model uses $(\mathbf{H} \bullet \mathbf{L})^s$. What is the geometric interpretation of these particular dot products? And, hence, what is the difference in the shape of the highlights between the Phong and Blinn Models.

5. [Total: 20 points] Ray Tracing.

Consider ray tracing in 2D. Derive an equation for the intersection of a ray in 2D with an ellipse $\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$.

SCORES:

1. Color (20)

2. Transformations (20)

3. Bezier curves (20)

4. Reflection models (20)

5. Ray Tracing (20)

TOTAL SCORE :

ANSWER SHEET
Comprehensive Examination in LOGIC
November 2000

MAGIC NUMBER: _____

1		2		3		4		5	
6		7		8		9		10	
11		12		13		14		15	
16		17		18		19		20	
21		22		23		24		25	
26		27		28		29		30	
31		32		33		34		35	
36		37		38		39		40	
41		42							

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.

I acknowledge and accept the Honor Code. (Signed) _____

Comprehensive Examination in Logic
Stanford University
Department of Computer Science
November 2000

Instructions

Please read these instructions and the *Notation* section carefully. Do not read beyond this page until instructed to do so.

You should mark your ansers only in the **answer sheet** that is provided with this part of the Comprehensive Examination. Be sure to write your **magic number** on the answer sheet.

This exam is **open book** and is composed of **42 questions** on **7 pages**, plus one answer sheet. For each question, write either **YES** or **NO** in the corresponding box of the answer sheet, or leave it blank. You will receive **+2** points for each correct answer, **−3** points for each incorrect answer, and **0** points for a blank (or crossed out) answer. You have **60 minutes** to complete the exam.

Notation

The notation is the one used by Enderton in *A Mathematical Introduction to Logic*, with the difference that the equality symbol is denoted by $=$ instead of $\approx$ and arguments to predicate and function symbols are enclosed in parentheses and separated by commas. Thus, for example, instead of Enderton's $fxyz$, $f(x, y, z)$ is used.

In some problems, the following symbols are used, whose definition is repeated here for completeness:

- $\mathrm{Cn}(A)$ is the set of consequences of an axiom set A;
- $\mathrm{Th}(\mathfrak{M})$ is the first-order theory of the structure $\mathfrak{M}$, i.e. the set of first-order sentences, of a given language, that are true in $\mathfrak{M}$.

Do not turn this page until instructed to do so.

Propositional Logic

Which of the following are complete sets of connectives?

1. $\wedge, \vee$
2. $\rightarrow, \neg$

If P means *"toves are slithy"* and Q means *"borogoves are mimsy"*, which of the following formulas mean *"toves are not slithy, unless borogoves are mimsy"*?

3. $\neg P \rightarrow Q$
4. $P \rightarrow Q$
5. $Q \rightarrow P$

Predicate Logic

Which of the following is a valid sentence of first-order logic?

6. $(\forall x\ P(x)) \rightarrow (\exists y\ P(y))$
7. $(\exists x\ P(x)) \rightarrow (\forall y\ P(y))$
8. $\exists x\ (P(x) \rightarrow \forall y\ P(y))$
9. $\forall x\ (\neg(x = 0) \rightarrow \exists y\ x = \mathrm{S}(y))$

Unification

Which of the following are true about unification in first-order logic?

10. $\{x \leftarrow z, y \leftarrow f(z)\}$ is an m.g.u. (most general unifier) of $f(g(x), y)$ and $f(g(y), f(x))$.
11. $\{y \leftarrow f(x)\}$ is an m.g.u. of $f(f(x), y)$ and $f(y, f(x))$.
12. $\{x \leftarrow y, y \leftarrow f(y)\}$ is an m.g.u. of $f(f(x), y)$ and $f(y, f(x))$.
13. Let θ be a unifier of t_1 and t_2. Then $\langle t_1, t_2, t_3 \rangle$ are unifiable if and only if $t_1\theta$ and $t_3\theta$ are unifiable.

Skolemization

In the following, "to skolemize" means to skolemize preserving validity, as in *The Deductive Foundations of Computer Programming*. x, x_1, x_2, y, z are variables.

14. Is the existential closure of $\neg P(x_1, y) \rightarrow \neg P(x_2, f(x_2))$ a correct skolemization of $(\neg\forall x\, \exists y\ P(x, y)) \rightarrow (\exists x\, \forall y\ \neg P(x, y))$?

15. Is the existential closure of $\neg(P(z, y) \rightarrow P(z, f(z)))$ a correct skolemization of $\exists y\ \neg\forall z\ (P(z, y) \rightarrow \exists y\ P(z, y))$?

Deductive Tableaux

Consider the following deductive tableau:

	A	G
1	$P(x, f(x)) \wedge Q(y) \rightarrow P(f(y), f(f(y)))$	
2	$P(f(a), f(f(a))) \vee Q(a)$	
3		$P(x, f(x))$

Which of the following rows can be added to the tableau by one correct application of a resolution rule?

16.

4	$\neg Q(a)$	

17.

4		$\neg Q(a)$

18.

4	$P(x, f(a)) \wedge Q(a)$	

19.

4		$P(x, f(a)) \wedge Q(a)$

Given a deductive tableau $\mathcal{T}$, for a first-order logic, in which there is no occurrence of the equality symbol, quantifiers, $\top$ or $\bot$, which of the following are true?

20. If an assertion and a goal in $\mathcal{T}$ are unifiable, then $\mathcal{T}$ is valid.

21. If no resolution rule can be applied, then $\mathcal{T}$ is not valid.

22. There exists a tableau containing only assertions (no goals) to which $\mathcal{T}$ is equivalent.

Let A be a finite set of axioms for a theory $T = \mathrm{Cn}(A)$, over a language with equality, φ a formula in the same language. Which of the following are then necessarily true?

23. If, starting from a tableau containing only formulas in A as assertions and only φ as goal, after a finite number of applications of resolution, quantifier elimination, and equality rules one gets a tableau with $\top$ as a goal or $\bot$ as an assertion, then $T \vDash \varphi$.

24. Let SPO be the theory of strict partial orderings (over the language with the binary predicate symbol $\prec$ and no equality) given by the two axioms tr (for *"transitivity"*) and ir (for *"irreflexivity"*), i.e. $SPO = \mathrm{Cn}(\{tr, ir\})$. Is it true, then, that a necessary and sufficient condition for $SPO \vDash \varphi$ is the existence of a tableau proof starting from the initial tableau

1.		$\neg tr \vee \neg ir \vee \varphi$	?

Polarity

Let A be a set of first-order sentences and A' be obtained from A by replacing every occurrence of a P-atom of positive polarity by $\top$. (A P-atom is an atomic formula of the form $P(\dots)$, where P is a predicate symbol of arbitrary arity.) Let $T = \mathrm{Cn}(A)$ and $T' = \mathrm{Cn}(A')$. Which of the following hold?

25. If $T \vDash \varphi$, then $T' \vDash \varphi$.

26. If $\mathfrak{M}$ is a model for T, then $\mathfrak{M}$ is a model for T'.

First-Order Theories

Which of the following are decidable?

27. The set of proofs in the language $(\mathbf{0}, \mathbf{S}, \mathbf{+}, \cdot)$.

28. The set of sentences true in the structure $(\mathbb{N}, 0, S, +)$.

29. The set of sentences valid in the first-order logic of the language $(\mathbf{0}, \mathbf{S}, \mathbf{+}, \cdot)$.

Which of the following are true? ($\mathfrak{N}$ is the structure $(\mathbb{N}, 0, S, <, +, \cdot, E)$, i.e. the standard model of natural numbers; E is the exponentiation function)

30. All countable models of $\mathrm{Th}(\mathfrak{N})$ are elementarily equivalent.

31. All countable models of $\mathrm{Th}(\mathfrak{N})$ are isomorphic.

32. $\mathrm{Th}(\mathfrak{N})$ is complete.

33. $\mathrm{Th}(\mathfrak{N})$ is recursively enumerable.

Consider a first-order language with one binary predicate symbol R and equality. Which of the following hold in this language?

34. There is a satisfiable formula all whose models are finite.

35. There is a satisfiable formula all whose models are infinite.

36. There is a satisfiable formula all whose models are countably infinite.

Consider a first-order language with equality, one binary predicate symbol R, and no other parameters. Furthermore, consider the theory $T = \mathrm{Cn}(A)$ of all logical consequences of axiom set A:

$$\forall x \forall y \forall z \ (x = y \vee y = z \vee x = z)$$
$$\forall x \ R(x, x)$$

37. Is T complete?

38. Is T decidable?

39. Is T recursively enumerable?

40. Is T axiomatizable?

Well-Founded Induction

Which of the following relations are well-founded?

41. On tuples (as defined in *The Deductive Foundations of Computer Programming* , the relation R defined by

$$R(x, y) \leftrightarrow x = \mathit{tail}(y) \ .$$

42. On non-negative integers, the relation R defined by

$$\begin{aligned} R(x, y) \leftrightarrow & \exists z \ (\ y = \mathbf{S}(\mathbf{S}(\mathbf{0})) \cdot z \ \wedge \ x + \mathbf{S}(\mathbf{0}) = z \ ) \\ & \vee \exists z \ (\ (\mathbf{S}(\mathbf{S}(\mathbf{0})) \cdot z) + \mathbf{S}(\mathbf{0}) = y \ \wedge \ x = z + \mathbf{S}(\mathbf{0}) \ ) \ . \end{aligned}$$

Comprehensive Examination in Logic
Stanford University
Department of Computer Science
November 2000

Solutions

Instructions

Please read these instructions and the *Notation* section carefully. Do not read beyond this page until instructed to do so.

You should mark your ansers only in the **answer sheet** that is provided with this part of the Comprehensive Examination. Be sure to write your **magic number** on the answer sheet.

This exam is **open book** and is composed of **42 questions** on **12 pages**, plus one answer sheet. For each question, write either **YES** or **NO** in the corresponding box of the answer sheet, or leave it blank. You will receive **+2** points for each correct answer, **−3** points for each incorrect answer, and **0** points for a blank (or crossed out) answer. You have **60 minutes** to complete the exam.

Notation

The notation is the one used by Enderton in *A Mathematical Introduction to Logic*, with the difference that the equality symbol is denoted by $=$ instead of $\approx$ and arguments to predicate and function symbols are enclosed in parentheses and separated by commas. Thus, for example, instead of Enderton's $fxyz$, $f(x, y, z)$ is used.

In some problems, the following symbols are used, whose definition is repeated here for completeness:

- $\mathrm{Cn}(A)$ is the set of consequences of an axiom set A;
- $\mathrm{Th}(\mathfrak{M})$ is the first-order theory of the structure $\mathfrak{M}$, i.e. the set of first-order sentences, of a given language, that are true in $\mathfrak{M}$.

Do not turn this page until instructed to do so.

Propositional Logic

Which of the following are complete sets of connectives?

1. $\wedge, \vee$

Answer. NO. It can be proved by induction on formulas build from these connectives that every such formula is true under the truth assignment that assigns a true value to every propositional symbol. Thus, for example, no contradiction can be written in this language.

2. $\rightarrow, \neg$

Answer. YES. $\varphi \vee \psi$ is equivalent to $\neg\varphi \rightarrow \psi$, $\varphi \wedge \psi$ is equivalent to $\neg(\varphi \rightarrow \neg\psi)$, etc.

If P means *"toves are slithy"* and Q means *"borogoves are mimsy"*, which of the following formulas mean *"toves are not slithy, unless borogoves are mimsy"*?

3. $\neg P \rightarrow Q$

Answer. NO.

4. $P \rightarrow Q$

Answer. YES.

5. $Q \rightarrow P$

Answer. NO.

Predicate Logic

Which of the following is a valid sentence of first-order logic?

6. $(\forall x\ P(x)) \rightarrow (\exists y\ P(y))$

Answer. YES. This depends on the assumption that the domain of a first-order interpretation cannot be the empty set. Alternatively, the formula can be proved in your favorite calculus.

7. $(\exists x\ P(x)) \rightarrow (\forall y\ P(y))$

Answer. NO. As a counterexample, take an interpretation with a two-element domain, P holding of one only.

8. $\exists x\ (P(x) \rightarrow \forall y\ P(y))$

Answer. YES. This is known as *Beth's formula*. Given any interpretation, there are two cases: either $\forall y\ P(y)$ is true, and then the implication is always true and any value for x will do, or it is false, and then there is an element of the domain for which P does not hold, and one can assign that element to x, making the antecedent of the implication always false, hence the whole implication true.

9. $\forall x\ (\neg(x = \mathbf{0}) \rightarrow \exists y\ x = \mathbf{S}(y))$

Answer. NO. This is valid in models of arithmetic, but is not a validity of first-order logic. For a counterexample, take a two-element domain $\{a, b\}$, map $\mathbf{0}$ to a, let $\mathbf{S}$ be interpreted as the identity function returning a.

UNIFICATION

Which of the following are true about unification in first-order logic?

10. $\{x \leftarrow z, y \leftarrow f(z)\}$ is an m.g.u. (most general unifier) of $f(g(x), y)$ and $f(g(y), f(x))$.

Answer. NO. The two terms are not unifiable. To see this, apply the unification algorithm and you will end with an occur-check violation (an equation of the form $x = t$, with x occurring in t).

11. $\{y \leftarrow f(x)\}$ is an m.g.u. of $f(f(x), y)$ and $f(y, f(x))$.

Answer. YES. It is one of the possible solutions returned by the unification algorithm.

12. $\{x \leftarrow y, y \leftarrow f(y)\}$ is an m.g.u. of $f(f(x), y)$ and $f(y, f(x))$.

Answer. YES. All m.g.u.s of a term differ by a permutation of variables, and this can be obtained by the one in the previous problem by composition with the permutation $\{x \leftarrow y, y \leftarrow x\}$. Note that this m.g.u. cannot be returned by any execution of the unification algorithm, because it is not idempotent.

13. Let θ be a unifier of t_1 and t_2. Then $\langle t_1, t_2, t_3 \rangle$ are unifiable if and only if $t_1\theta$ and $t_3\theta$ are unifiable.

Answer. NO. θ needs to be most general for this to hold. Consider $t_1 = f(x)$, $t_2 = f(y)$, $t_3 = f(g(z))$, $\theta = \{x \leftarrow f(x), y \leftarrow f(x)\}$. θ is not most general and $\langle t_1\theta, t_3\theta \rangle$ is not unifiable, but $\langle t_1, t_2, t_3 \rangle$ clearly is.

Skolemization

In the following, "to skolemize" means to skolemize preserving validity, as in *The Deductive Foundations of Computer Programming.* x, x_1, x_2, y, z are variables.

14. Is the existential closure of $\neg P(x_1, y) \rightarrow \neg P(x_2, f(x_2))$ a correct skolemization of $(\neg\forall x\, \exists y\ P(x, y)) \rightarrow (\exists x\, \forall y\ \neg P(x, y))$?

Answer. NO. The occurrence of x_1 should be replaced by a constant.

15. Is the existential closure of $\neg(P(z, y) \rightarrow P(z, f(z)))$ a correct skolemization of $\exists y\ \neg\forall z\ (P(z, y) \rightarrow \exists y\ P(z, y))$?

Answer. NO. A correct skolemization would be $\neg(P(z, y) \rightarrow P(z, f(y, z)))$.

Deductive Tableaux

Consider the following deductive tableau:

	A	G
1	$P(x, f(x)) \wedge Q(y) \rightarrow P(f(y), f(f(y)))$	
2	$P(f(a), f(f(a))) \vee Q(a)$	
3		$P(x, f(x))$

Which of the following rows can be added to the tableau by one correct application of a resolution rule?

16.

4	$\neg Q(a)$	

Answer. NO.

17.

4		$\neg Q(a)$

Answer. YES. Resolve 2 and 3 according to the polarity strategy.

18.

4	$P(x, f(a)) \wedge Q(a)$	

Answer. NO.

19.

4		$P(x, f(a)) \wedge Q(a)$

Answer. NO.

The complete set of rows that can be inferred from the given ones by one application of a resolution rule is: goals $\bot$, $P(x, f(x)) \wedge Q(y)$, $\neg Q(a)$ and assertions $\top$, $(Q(y) \rightarrow P(f(y), f(f(y)))) \vee Q(a)$, $(P(x, f(x)) \rightarrow P(f(a), f(f(a)))) \vee P(f(a), f(f(a)))$, $\neg(P(x, f(x)) \wedge Q(a)) \vee Q(a)$.

Given a deductive tableau $\mathcal{T}$, for a first-order logic, in which there is no occurrence of the equality symbol, quantifiers, $\top$ or $\bot$, which of the following are true?

20. If an assertion and a goal in $\mathcal{T}$ are unifiable, then $\mathcal{T}$ is valid.

Answer. YES. A resolution step can be applied and yield the true goal in one step.

21. If no resolution rule can be applied, then $\mathcal{T}$ is not valid.

Answer. YES. With no quantifiers there is no need for skolemization, and with no equality the resolution rule alone is complete for validity.

22. There exists a tableau containing only assertions (no goals) to which $\mathcal{T}$ is equivalent.

Answer. YES. Just move all goals in $\mathcal{T}$ to assertion by prepending a $\neg$. According to the Duality Proposition, the two tableaux are equivalent.

Let A be a finite set of axioms for a theory $T = \mathrm{Cn}(A)$, over a language with equality, φ a formula in the same language. Which of the following are then necessarily true?

23. If, starting from a tableau containing only formulas in A as assertions and only φ as goal, after a finite number of applications of resolution, quantifier elimination, and equality rules one gets a tableau with $\top$ as a goal or $\bot$ as an assertion, then $T \models \varphi$.

Answer. YES. This is indeed a sufficient condition for validity. It is not necessary, though, since in general one must add the assertion $x = x$ to have completeness in presence of equality.

24. Let SPO be the theory of strict partial orderings (over the language with the binary predicate symbol $\prec$ and no equality) given by the two axioms tr (for *"transitivity"*) and ir (for *"irreflexivity"*), i.e. $SPO = \mathrm{Cn}(\{tr, ir\})$. Is it true, then, that a necessary and sufficient condition for $SPO \models \varphi$ is the existence of a tableau proof starting from the initial tableau

1.		$\neg tr \vee \neg ir \vee \varphi$

?

Answer. YES. The formula given as a goal is equivalent to ψ: $tr \wedge ir \rightarrow \varphi$. The existence of a tableau proof starting from here is equivalent to the validity of *psi*, which in turn is equivalent to the validity of φ in SPO.

Polarity

Let A be a set of first-order sentences and A' be obtained from A by replacing every occurrence of a P-atom of positive polarity by $\top$. (A P-atom is an atomic formula of the form $P(\ldots)$, where P is a predicate symbol of arbitrary arity.) Let $T = \mathrm{Cn}(A)$ and $T' = \mathrm{Cn}(A')$. Which of the following hold?

25. If $T \models \varphi$, then $T' \models \varphi$.

Answer. NO. For a simple counterexample, take φ to be $\forall x\ P(x)$ and A to be $\{\varphi\}$. The converse implication is true, as can be inferred from the next problem.

26. If $\mathfrak{M}$ is a model for T, then $\mathfrak{M}$ is a model for T'.

Answer. YES. For every axiom $\varphi \in A$ let φ' be the corresponding axiom in A'. Then, by the Polarity Proposition, $phi \longrightarrow \varphi'$. For $\mathfrak{M}$ to be a model of T means, by definition, that $\mathfrak{M} \models \varphi$ for all $\varphi in A$ and hence by the previous observation it follows that $\mathfrak{M} \models \varphi'$. Again by definition of being a model this means that $\mathfrak{M}$ is a model of T'.

First-Order Theories

Which of the following are decidable?

27. The set of proofs in the language $(\mathbf{0}, \mathbf{S}, +, \cdot)$.

Answer. YES. It can be checked algorithmically whether a given syntactic object is a correct proof. In general, any reasonable definition of proof must have this property.

28. The set of sentences true in the structure $(\mathbb{N}, 0, S, +)$.

Answer. YES. This is a substructure of Presburger Arithmetic.

29. The set of sentences valid in the first-order logic of the language $(\mathbf{0}, \mathbf{S}, +, \cdot)$.

Answer. NO. This is known as Church's Theorem.

Which of the following are true? ($\mathfrak{N}$ is the structure $(\mathbb{N}, 0, S, <, +, \cdot, E)$, i.e. the standard model of natural numbers; E is the exponentiation function)

30. All countable models of $\mathrm{Th}(\mathfrak{N})$ are elementarily equivalent.

Answer. YES. This holds in general for any structure $\mathfrak{M}$, since then $\mathrm{Th}(\mathfrak{N})$ is complete.

31. All countable models of $\mathrm{Th}(\mathfrak{N})$ are isomorphic.

Answer. NO. Add a new constant c to the language, and add to $\mathrm{Th}(\mathfrak{N})$ the axioms $\mathbf{0} < c$, $\mathbf{S(0)} < c$, $\mathbf{S(S(0))} < c$, *etc.* The resulting theory is finitely satisfiable, hence satisfiable by the Compactness Theorem. The restriction of a model of it to the language without c is a model of $\mathrm{Th}(\mathfrak{N})$ that is not isomorphic to $\mathfrak{N}$.

32. $\mathrm{Th}(\mathfrak{N})$ is complete.

Answer. YES. This holds for any structure $\mathrm{Th}(\mathfrak{N})$.

33. $\mathrm{Th}(\mathfrak{N})$ is recursively enumerable.

Answer. NO. This theory is undecidable.

Consider a first-order language with one binary predicate symbol R and equality. Which of the following hold in this language?

34. There is a satisfiable formula all whose models are finite.

Answer. YES. Take $\forall x\ x = x$.

35. There is a satisfiable formula all whose models are infinite.

Answer. YES. The standard example is

$$\begin{gathered}\forall x\ \neg R(x,x) \wedge \\ \forall x \forall y \forall z\ (R(x,y) \wedge R(y,z) \rightarrow R(x,z)) \wedge \\ \forall x \exists y\ R(x,y)\ .\end{gathered}$$

36. There is a satisfiable formula all whose models are countably infinite.

Answer. NO. This would violate the Löwenheim-Skolem Theorem.

Consider a first-order language with equality, one binary predicate symbol R, and no other parameters. Furthermore, consider the theory $T = \mathrm{Cn}(A)$ of all logical consequences of axiom set A:

$$\forall x \forall y \forall z \ (x = y \vee y = z \vee x = z)$$
$$\forall x \ R(x, x)$$

37. Is T complete?

Answer. NO. As a counterexample, neither the sentence $\forall x \forall y \ (R(x, y) \rightarrow R(y, x))$ nor its negation are valid in the theory. To reason about this problem, you should realize that the models can be thought as being the graphs with at most three nodes and having reflexive loops.

38. Is T decidable?

Answer. YES. One simply has to check a finite set of graphs and see whether a given sentence holds in all of them.

39. Is T recursively enumerable?

Answer. YES. Any theory with a recursively enumerable set of axioms is recursively enumerable.

40. Is T axiomatizable?

Answer. YES. A (finite, hence recursive) set of axioms is given by the problem.

Well-Founded Induction

Which of the following relations are well-founded?

41. On tuples (as defined in *The Deductive Foundations of Computer Programming* , the relation R defined by

$$R(x, y) \leftrightarrow x = \mathit{tail}(y) \ .$$

Answer. YES. If $R(x, y)$, then x is a strictly shorter tuple than y,and hence there cannot be an infinite descending chain.

42. On non-negative integers, the relation R defined by

$$\begin{aligned} R(x, y) \leftrightarrow & \exists z \ (\ y = \mathbf{S}(\mathbf{S}(\mathbf{0})) \cdot z \ \wedge \ x + \mathbf{S}(\mathbf{0}) = z \ ) \\ & \vee \exists z \ (\ (\mathbf{S}(\mathbf{S}(\mathbf{0})) \cdot z) + \mathbf{S}(\mathbf{0}) = y \ \wedge \ x = z + \mathbf{S}(\mathbf{0}) \ ) \ . \end{aligned}$$

Answer. NO. In fact, this relation has a reflexive loop at 1. The whole point here was to see if you are acquainted with the language of first-order logic and can decrypt it fast enough.

ANSWER SHEET
Comprehensive Examination in LOGIC
November 2000

MAGIC NUMBER: _____

1	NO	2	YES	3	NO	4	YES	5	NO
6	YES	7	NO	8	YES	9	NO	10	NO
11	YES	12	YES	13	NO	14	NO	15	NO
16	NO	17	YES	18	NO	19	NO	20	YES
21	YES	22	YES	23	YES	24	YES	25	NO
26	YES	27	YES	28	YES	29	NO	30	YES
31	NO	32	YES	33	NO	34	YES	35	YES
36	NO	37	NO	38	YES	39	YES	40	YES
41	YES	42	NO						

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.

I acknowledge and accept the Honor Code. (Signed) _____

NET

Computer Science Department

Stanford University

Comprehensive Examination in Networks

Fall 2000

READ THIS FIRST!!

1. You should write your answers for this part of the Comprehensive Examination in a BLUE BOOK. Be sure to write your MAGIC NUMBER on the cover of every blue book you use.
2. The number of POINTS for each problem indicates how elaborate an answer should be. For example, a question worth 6 points or less doesn't deserve an extremely detailed answer, even if you feel you could expound at length upon it. **Short, bulleted answers are encouraged.**
3. The total number of points is 60.
4. The exam is CLOSED BOOK. You may NOT use notes, books, computers, other people, etc.
5. Show your work, since PARTIAL CREDIT will be given for incomplete answers.
6. If you need to make an assumption to answer a question, state your assumption(s) as well as your answer.
7. Be sure to provide justification for your answers.

Problem 1 (15 points)

Wireless networks may suffer from interference and poor signal strength, causing packets to drop.

a. (3 points) Some wireless link layers will retransmit a dropped packet some finite number of times. What is the advantage to having the link layer perform such a service?
b. (6 points) Given such a service at the link layer, what, if any, are the reasons to run TCP over such a network?
c. (6 points) Some wireless devices will attempt to retransmit a dropped packet indefinitely (retrying it periodically in the midst of sending other newer packets. Is there any disadvantage to a link layer performing this service rather than leaving all retransmissions up to TCP? If a link layer knew TCP was running on top of it, how might it adapt itself?

Problem 2 (10 points)

Wireless access to the Internet is growing. These wireless access links, however, are often low-bandwidth. Some network architectures place proxies at the other end (from the user) of a wireless link, so that the proxy can perform "invisible" services on behalf of user to mitigate the effect of the low-bandwidth link. For instance, if a user asks for web pages from a server, the proxy may convert graphics on those pages to low-resolution, smaller, black&white pictures that consume little bandwidth. What issues arise if the data the user requests from the server is encrypted with the user's key, and what techniques can you come up with to address these issues?

Problem 3 (20 points)

The Border Gateway Protocol (BGP) is used to route packets between autonomous systems in the Internet.

a. (5 minutes) Why is a different routing protocol used between autonomous systems than is used within them? (The issues are not necessarily technical ones.)
b. (10 minutes) Instead of maintaining just a cost to each destination, as do most distance vector protocols, each BGP router keeps track of the exact path used. Each BGP router periodically tells its neighbors the exact path it is using to a destination. Why is this exact path information useful to BGP?
c. (5 minutes) Why do some autonomous networks decline to carry "transit traffic?" (Transit traffic is traffic that neither originates nor terminates in the autonomous system.)

Problem 4 (15 points)

Some wireless networks are unable to provide broadcast service. Does this affect any of the following services in the Internet, and if so, how? What could you do about it?

a. ARP
b. DHCP/BOOTP
c. DNS lookups

NET

Computer Science Department

Stanford University

Comprehensive Examination in Networks

SOLUTIONS

Fall 2000

READ THIS FIRST!!

1. You should write your answers for this part of the Comprehensive Examination in a BLUE BOOK. Be sure to write your MAGIC NUMBER on the cover of every blue book you use.
2. The number of POINTS for each problem indicates how elaborate an answer should be. For example, a question worth 6 points or less doesn't deserve an extremely detailed answer, even if you feel you could expound at length upon it. **Short, bulleted answers are encouraged.**
3. The total number of points is 60.
4. The exam is CLOSED BOOK. You may NOT use notes, books, computers, other people, etc.
5. Show your work, since PARTIAL CREDIT will be given for incomplete answers.
6. If you need to make an assumption to answer a question, state your assumption(s) as well as your answer.
7. Be sure to provide justification for your answers.

Problem 1 (15 points)

Wireless networks may suffer from interference and poor signal strength, causing packets to drop.

a. (3 points) Some wireless link layers will retransmit a dropped packet some finite number of times. What is the advantage to having the link layer perform such a service? The timeout for such retransmissions is typically quite small.

You don't have to incur the overhead of a TCP/transport-layer timeout.

b. (6 points) Given such a service at the link layer, what, if any, are the reasons to run TCP over such a network?

TCP handles additional kinds of functionality, for example:

1. *losses not in the network (e.g. losses in the host OS itself)*
2. *losses that exceed the link layer retransmit limit*
3. *losses as a result of congestion (i.e. packets dropped from queues)*

c. (6 points) Some wireless devices will attempt to retransmit a dropped packet indefinitely (retrying it periodically in the midst of sending other newer packets. Is there any disadvantage to a link layer performing this service rather than leaving all retransmissions up to TCP? If a link layer knew TCP was running on top of it, how might it adapt itself?

There are at least two disadvantages to this approach:

1. *it causes packets to be delivered out of order*
2. *it wastes bandwidth if TCP has already retransmitted the packet*

An ideal wireless link layer for TCP would avoid TCP coarse-grained timeouts and useless retransmissions and would keep packets in order by retransmitting a few times, but only within the time before TCP will retransmit.

Problem 2 (10 points)

Wireless access to the Internet is growing. These wireless access links, however, are often low-bandwidth. Some network architectures place proxies at the other end (from the user) of a wireless link, so that the proxy can perform "invisible" services on behalf of user to mitigate the effect of the low-bandwidth link. For instance, if a user asks for web pages from a server, the proxy may convert graphics on those pages to low-resolution, smaller, black&white pictures that consume little bandwidth. What issues arise if the data the user requests from the server is encrypted with the user's key, and what techniques can you come up with to address these issues?

The problem is that the proxy can't decrypt the server's data, so it cannot perform appropriate adaptations on it. One solution would be to give the proxy the user's keys, so the proxy can decrypt the data to perform various transformations on it, and then re-

encrypt the results. This presents some security concerns: the keys would need to be transferred to the proxy in a secure fashion, and they would need to be kept secure on the proxy.

Problem 3 (20 points)

The Border Gateway Protocol (BGP) is used to route packets between autonomous systems in the Internet.

a. (5 minutes) Why is a different routing protocol used between autonomous systems than is used within them? (The issues are not necessarily technical ones.)

Interior routing protocols need only maximize the efficiency of routing. External routing protocols must also deal with political issues, such as one company not being willing to allow its data to flow through its competitor's site.

b. (10 minutes) Instead of maintaining just a cost to each destination, as do most distance vector protocols, each BGP router keeps track of the exact path used. Each BGP router periodically tells its neighbors the exact path it is using to a destination. Why is this exact path information useful to BGP?

This allows a BGP router to avoid choosing paths that flow through itself, and it allows it to implement political decisions. To use the (rather naïve) example from above: by examining the path a packet would take if handed to a particular neighbor, the router can determine whether that neighbor might forward the packet to a competitor's site. If so, it could pick a different direction for the packet.

c. (5 minutes) Why do some autonomous networks decline to carry "transit traffic?" (Transit traffic is traffic that neither originates nor terminates in the autonomous system.)

Transit traffic performs no service for anyone inside the autonomous system, since it doesn't come from users there, and it isn't destined for users there. It nonetheless ties up bandwidth and other resources on the network.

Problem 4 (15 points)

Some wireless networks are unable to provide broadcast service. Does this affect any of the following services in the Internet, and if so, how? What could you do about it?

a. ARP

Yes, since ARP requires a broadcast to find out the hardware address for an IP address on the subnet. A possible solution is to make the address of an ARP server a well-known address, so ARP requests could be directed to a particular machine. This would mean some limited configuration information on hosts ahead of time, though, which is what ARP is attempting to avoid. A bit of an improvement might be to direct all ARP requests to the closest base station, and base stations would redirect the requests to an appropriate ARP server.

b. DHCP/BOOTP

Yes, since DHCP and BOOTP both require broadcasts to find out a host's own IP address upon initialization of the network on the host. A possible solution is to direct all requests to a well-known DHCP server. As above, the base station solution might be appropriate.

c. DNS lookups

No. The address of the DNS server is one of the pieces of information gained through whatever configuration process network initialization on a host uses. The actual lookups are directed to the DNS server and are not broadcast.

Computer Science Department
Stanford University
Comprehensive Examination in Numerical Analysis
Autumn 2000

Instructions :

- Answer all questions. Total 30 points.
- Marks for each question and its parts are specified.
- Books and notes are not permitted.
- Be brief and clear.

---

1. [10 marks] Solution of linear equations.
Suppose we want to solve the linear system $A\mathbf{x} = \mathbf{b}$.

   (a) Give a 2×2 example where Gaussian Elimination breaks down on the first step. [2 points]

   (b) Describe the partial pivoting strategy and the complete pivoting strategy. [5]

   (c) In what situation would one prefer partial pivoting, and when would one use complete pivoting? [3]

2. [8 marks] Iterative solution of linear equations.
Let A be an $n \times n$ matrix and $\mathbf{b}$ a given vector. We write $A = D + L + U$, where D is the diagonal of A, and L and U are the strictly lower and upper triangular parts of A, respectively.

   (a) Define the Jacobi and Gauss-Seidel iterations for solving the system of equations. [4]

   (b) Give a sufficient condition for the approximates to converge to the solution $\mathbf{x}$, independent of the initial guess. [4]

3. [12 marks] Ordinary differential equations.
Consider the equation

$$y'(x) = \lambda y(x),$$
$$y(0) = 1.$$

Suppose that λ is a real negative number.

(a) Give the exact analytic solution to the o.d.e and discuss the solution as $x \to \infty$. [2]

(b) Suppose we use the Forward Euler method:

$$y_{k+1} = y_k + hy'_k, \quad k = 0, 1, 2, \ldots$$

Under what conditions will $y_k \to 0$ as $k \to \infty$? [3]

(c) Consider now the *backward* Euler method:

$$y_{k+1} = y_k + hy'_{k+1}, \quad k = 0, 1, 2, \ldots$$

Under what conditions will $y_k \to 0$ as $k \to \infty$? [3]

(d) Consider now a *general* Initial Value Problem of the form $y' = f(x, y)$, with an initial value $y(x_0) = y_0$. Denote $x_k = x_0 + kh$, and suppose that y_k is the approximation to the solution $y(x_k)$, that is obtained by applying the forward Euler method. Give an estimate for

$$|y(kh) - y_k|.$$

[4]

3. [12 marks] Ordinary differential equations.
Consider the equation

$$y'(x) = \lambda y(x),$$
$$y(0) = 1.$$

Suppose that λ is a real negative number.

(a) Give the exact analytic solution to the o.d.e and discuss the solution as $x \to \infty$. [2]

(b) Suppose we use the Forward Euler method:

$$y_{k+1} = y_k + hy'_k, \quad k = 0, 1, 2, \ldots$$

Under what conditions will $y_k \to 0$ as $k \to \infty$? [3]

(c) Consider now the *backward* Euler method:

$$y_{k+1} = y_k + hy'_{k+1}, \quad k = 0, 1, 2, \ldots$$

Under what conditions will $y_k \to 0$ as $k \to \infty$? [3]

(d) Consider now a *general* Initial Value Problem of the form $y' = f(x, y)$, with an initial value $y(x_0) = y_0$. Denote $x_k = x_0 + kh$, and suppose that y_k is the approximation to the solution $y(x_k)$, that is obtained by applying the forward Euler method. Give an estimate for

$$|y(kh) - y_k|.$$

[4]

Comprehensive Exam — Programming languages

Fall 2000

Problem 1 — Activation records (8 points)
You want to modify your compiler to support dynamically-sized, stack-allocated variables. Assuming a traditional machine architecture, explain what complications this creates compared to the usual constant-size stack allocation, and how you would handle it.

Problem 2 — Execution (12 points)
One of CS's core themes is execution. We package it in various ways: procedure calls, threads, processes, co-routines, upcalls, interrupt handlers, etc.

1. **(6 points)** Ignoring hardware and language details, what do you fundamentally need to "execute" an instruction stream and why? Please pick two of the above abstractions and, at a high level, classify their constituent parts in your categories.

2. **(6 points)** Whenever we have multiple streams of execution, we have to decide what to which stream to run ("scheduling"). What are some differences between thread scheduling and scheduling which procedure to run? What property of procedure scheduling allows us to use a single stack for procedures but (in general) forces multiple stacks for threads?

Problem 3 — GC fun (18 points)
Assume you want to add garbage collection to C. Unfortunately, unlike most GC implementations, you do not have any compiler support. So, instead you do a hack where you treat all memory/register whose values are valid memory addresses as pointers.

1. (12 points) At a high level, explain how to write a (mostly) conservative garbage collector using this trick.

2. (6 points) When will your collector lose storage? Can you give (possibly) contrived examples of where it would reclaim storage that is not dead? Can your collector compact the heap?

Problem 4 — Naming fun (20 points)
Assume you add overloading to C, where there can be multiple functions with the same name but different type signatures and the compiler determines uses argument types to decide which function to call. For example, given the following two functions:

```
int abs(int x);
float abs(float x);
```

the following calls will be resolved to the first and second respectively:

```
int x1;
float x2;

x1 = abs(x1);
x2 = abs(x2);
```

Assume we want to use separate compilation, and linkers only know how to bind a reference to name N to a single definition of name N. (I.e., emitting both implementations of "`abs`" with the same name will cause a "multiple definition" error.)

1. **(12 points)** Explain how to implement overloading using only local analysis given this constraint. You can simplify your scheme by requiring callers and modules that contain definitions to obey *reasonable* restrictions. Make sure to mention how to (1) stop the user from accidently colliding with your strategy; (2) working in the presence of debugging; and (3) support structures as arguments.

2. **(8 points)** What rules should you follow for resolving ambiguities in the presence of argument promotion? (I.e., when an argument of type T can be legally changed to a T'.) Be sure to handle conflicts that can arise with multi-argument functions.

SYS

Computer Science Department

Stanford University

Comprehensive Examination in Software Systems
Fall 2000

Read This First!

1. Write **all** answers in a blue book. **No credit** is given for answers written on these exam pages. *Don't panic!* This seems long, but most of the answers are very short.

2. Be sure to write your **MAGIC NUMBER** on the cover of **EACH** blue book you use.

This is an **OPEN BOOK** exam. You can't look up stuff on the Internet or ask other people, but you may use any books, notes, etc. you like, as well as a non-Internet-connected computer.

3. Use the point values of each question or subquestion to help plan your time. *Read the whole exam first; in case it's too long, do the ones you know right away.*

4. Justify your answers! Partial credit is given for good reasoning but a wrong answer, whereas it won't be given for a correct answer with incorrect or no reasoning. *State clearly* any additional assumptions you make.

1 Miscellaneous topics, concise answers [13]

a) [3] You compile the following code in C using a typical Unix or Windows C compiler. Suppose you run it and call the function *func* with *x* set to –1 (negative one). What happens and why?

```
void func(int x)
{
   int a[10],b[10],c[10];
  ...code to initialize all elements of a[] to 1,b[] to 2,c[] to 3...
   printf("%d\n", b[x]);
}
```

b) [3] Alice emails her stockbroker the following string in an email message: "Sell 100 shares of Oracle!" She encrypts and digitally signs the message so he knows it's authentic. Unknown to her, the evil Mitch is sniffing the email server. How can Mitch attack Alice? What could she have done to prevent it?

c) [3] In a remote procedure call (RPC) system, one possible problem is that it is impossible to distinguish between a really slow callee and a failed callee. Why might it be a bad idea to just retry the call after a certain amount of time has expired? (Ignore the possibility of overloading the network or RPC server.)

d) [4] A simple multithreaded program features *N* threads that share access to a common integer value *x*. To assure that only one thread at a time writes *x* (and that changes aren't lost), each thread's critical section looks like this:

```
/* begin critical section */
Lock(x);     /* will spin if necessary until lock is available */
x = newIntegerValue;
Unlock(x);
/* end critical section */
```

Assume you have an instruction *CompareAndSwap2*, which is similar to the familiar atomic *CompareAndSwap*. CAS2 performs the following operation *atomically:*

```
int CAS2(int *x, int *y, int oldXval, int newXval, int newYval)
{
   if (*x == oldXval) {
      *x = newXval;
      *y = newYval;
      return SUCCESS;
   } else {
      /* don't change x or y */
      return FAILURE;
}
```

Use CAS2 to rewrite the pseudocode for each thread's critical section *without using locks.* (Hint: use versioning.)

2 Memory Fragmentation [11]

Your superwizzy C libraries and runtime system provide standard system calls to allocate and free chunks of memory. There are no *a priori* restrictions on the size or alignment of memory that can be requested. The C function prototypes are roughly as follows:

```
typedef void *MemPtr;
MemPtr PtrAlloc(unsigned long sizeInBytes);
      /* PtrAlloc returns the NULL pointer if request can't be satisfied */
void PtrFree(MemPtr aPtr);
```

a) [3] Briefly describe the *memory fragmentation* problem and how it might cause a MemAlloc() request to fail even if there is enough unused memory to satisfy the request.

To alleviate this problem, you modify your runtime system and C libraries to support *handles*. A handle is a double-indirection to a block of memory:

```
typedef MemPtr *MemHandle;
      /* you can also think of it as: typedef void **MemHandle  */
MemHandle HndAlloc(unsigned long sizeInBytes);
void HandleFree(MemHandle aHandle);
```

b) [4] Explain how handles alleviate the fragmentation problem.

c) [4] Describe *two* performance impacts that arise when programmers routinely use handles. (**Hint:** one occurs frequently, the other relatively infrequently.)

3 Filesystems and Leases [12]

You're designing an NFS-like network file system for Unix that allows many clients to access and edit files on a network-connected remote fileserver. We'll refer to the server as *S* and three clients as *X,Y,Z*. When a client opens a file, the server sends a copy of the *whole* file to the client. In addition:

1. Any number of clients may simultaneously open a file for *read only* access.

2. If X opens the file for writing, a lock is set on the server so that future clients can *only* open that file for reading. When X closes the file, the lock is released so that future clients may open the file for writing. The entire act of writing X's changes and releasing the lock is atomic with respect to the server. The new contents of the file are *not* automatically sent to clients that have the file open for reading.

a) [3] Suppose X obtains a write lock on a file, and then crashes. When X reboots itself, it "forgets" which file(s) it had write locks on. What is the effect of this failure on the other clients and on the server?

To remedy the problem of (a) , you suggest that the server use *leases*. A lease gives X the right to access the file *for a limited amount of time*. (As before, at most one client can hold a write lock on the file.) When that time expires, if X still wants to use the file, it must ask the server to renew the lease, otherwise the server will unilaterally terminate the lease (and release the write lock, if the leaseholder had one).

b) [4] Explain how leases fix the problem in the scenario of part (a) , and explain any new effects seen by X in that scenario after it reboots.

c) [3] Suppose X is more careful: when it obtains a lease, it also records the fact that it is editing a particular file, and locally saves changes to that file as edits are in progress. X now crashes, reboots, and allows the user to recover the *local copy* of the file she was editing. Describe a scenario and the circumstances under which X might perceive a filesystem inconsistency.

d) [2] Assume that you can guarantee that any client's recovery time after a crash is at most R. Describe one possible way to avoid the inconsistency of part (c), and describe its effect on the system.

4 Debugging Breakpoints [15]

You have been assigned the job of adding *data breakpoints* to an existing C debugger. The desired behavior is that the programmer can "mark" particular variables as being breakpoints: whenever a marked variable is *read or modified*, the debugger should take a breakpoint and allow the programmer to inspect the program's state, etc., then resume execution.

Assume that you are *not* allowed to make the programmer modify or recompile her source code, but you *do* have full access to the operating system and the runtime system, and in particular you can modify the virtual memory functions of the operating system (page fault handlers, page tables, etc.).

Also assume (as is the case in most implementations) that the compiler and linker arrange to store *global* variables in a designated memory pages that is known at link time, that all global variables will fit on one page, and that that page is not used for storing anything other than global variables.

a) [4] Suppose first that we only care about being able to mark global variables. Explain in detail how you would use the OS's virtual memory system to implement data breakpoints. In your explanation, keep in mind that only *some* global variables are likely to be marked.

b) [4] Qualitatively describe the impact on the overall speed of execution when the programmer marks a variable. What factor(s) dominate this impact? To what extent does the *number* of marked variables influence performance (assuming all marked variables are referenced equally often)?

c) [3] Suppose we instead want to support only *modify* breakpoints: a marked variable should cause a breakpoint only when its value is *modified*, not when it is read. What modifications could you make to your implementation to support modify-breakpoints more efficiently than read-breakpoints?

d) [4] Describe what additional complication(s) you would encounter if you also had to implement this feature for functions' local variables. Identify *at least one* important factor that would affect performance if this feature is added, above and beyond the performance effects already discussed.

545

Computer Science Department

Stanford University

Comprehensive Examination in Software Systems
Fall 2000

SOLUTIONS

Read This First!

1. Write **all** answers in a blue book. **No credit** is given for answers written on these exam pages. *Don't panic!* This seems long, but most of the answers are very short.

2. Be sure to write your **MAGIC NUMBER** on the cover of **EACH** blue book you use.

This is an **OPEN BOOK** exam. You can't look up stuff on the Internet or ask other people, but you may use any books, notes, etc. you like, as well as a non-Internet-connected computer.

3. Use the point values of each question or subquestion to help plan your time. *Read the whole exam first; in case it's too long, do the ones you know right away.*

4. Justify your answers! Partial credit is given for good reasoning but a wrong answer, whereas it won't be given for a correct answer with incorrect or no reasoning. *State clearly* any additional assumptions you make.

1 Miscellaneous topics, concise answers [13]

a) [3] You compile the following code in C using a typical Unix or Windows C compiler. Suppose you run it and call the function *func* with x set to –1 (negative one). What happens and why?

```
void func(int x)
{
   int a[10],b[10],c[10];
  ...code to initialize all elements of a[] to 1,b[] to 2,c[] to 3...
   printf("%d\n", b[x]);
}
```

Answer. Depending on the implementation of the compiler, this will print either "1" or (more likely) "3". *No* implementation will throw an exception or give a memory error unless it stores automatic variables non-contiguously on the stack.

b) [3] Alice emails her stockbroker the following string in an email message: "Sell 100 shares of Oracle!" She encrypts and digitally signs the message so he knows it's authentic. Unknown to her, the evil Mitch is sniffing the email server. How can Mitch attack Alice? What could she have done to prevent it?

Answer. Mitch can replay the message later and cause another 100 shares to be sold; the broker can't distinguish Alice's message from Mitch's on the basis of content. Alice should have placed a timestamp or nonce (unique identifier) in her original message.

c) [3] In a remote procedure call (RPC) system, one possible problem is that it is impossible to distinguish between a really slow callee and a failed callee. Why might it be a bad idea to just retry the call after a certain amount of time has expired? (Ignore the possibility of overloading the network or RPC server.)

Answer. The function being performed by the RPC might not be idempotent (e.g. might have side effects). Without knowing if the first call succeeded, it's not necessarily safe to retry.

d) [4] A simple multithreaded program features N threads that share access to a common integer value x. To assure that only one thread at a time writes x (and that changes aren't lost), each thread's critical section looks like this:

```
/* begin critical section */
Lock(x);      /* will spin if necessary until lock is available */
x = newIntegerValue;
Unlock(x);
/* end critical section */
```

Assume you have an instruction *CompareAndSwap2*, which is similar to the familiar atomic *CompareAndSwap*. CAS2 performs the following operation *atomically:*

```
int CAS2(int *x, int *y, int oldXval, int newXval, int newYval)
{
   if (*x == oldXval) {
      *x = newXval;
      *y = newYval;
      return SUCCESS;
   } else {
      /* don't change x or y */
      return FAILURE;
}
```

Use CAS2 to rewrite the pseudocode for each thread's critical section *without using locks*. (Hint: use versioning.)

Answer.

```
/* let x be the variable to be atomically changed, v its version number */
do {
  tempV = v;
} until (CAS2(&v, &x, tempV, tempV+1, newIntegerValue) == SUCCESS);
```

Clarification. Note that we're not really exploiting CAS2 very well here (and for that matter, not exploiting locks very well in the original pseudocode); it's more useful when you want to atomically change a variable *from one known value to another* in the presence of possible race conditions. Because of this ambiguity, if you wrote correct code that *does* use locks, you still got partial credit; if your code simply would not give correct results (with or without locks), you got no credit.

2 Memory Fragmentation [11]

Your superwizzy C libraries and runtime system provide standard system calls to allocate and free chunks of memory. There are no *a priori* restrictions on the size or alignment of memory that can be requested. The C function prototypes are roughly as follows:

```
typedef void *MemPtr;
MemPtr PtrAlloc(unsigned long sizeInBytes);
      /* PtrAlloc returns the NULL pointer if request can't be satisfied */
void PtrFree(MemPtr aPtr);
```

a) [3] Briefly describe the *memory fragmentation* problem and how it might cause a MemAlloc() request to fail even if there is enough unused memory to satisfy the request.

Answer. This problem arises when, through a sequence of allocations and deallocations, there is not enough *contiguous* memory to satisfy a request, even if there's enough total free memory.

To alleviate this problem, you modify your runtime system and C libraries to support *handles*. A handle is a double-indirection to a block of memory:

```
typedef MemPtr *MemHandle;
      /* you can also think of it as: typedef void **MemHandle  */
MemHandle HndAlloc(unsigned long sizeInBytes);
void HandleFree(MemHandle aHandle);
```

b) [4] Explain how handles alleviate the fragmentation problem.

Answer. When a memory allocation request comes, the OS can shuffle around the blocks in the heap and modify the second-level indirect pointers accordingly. As long as applications always use double indirection to dereference memory, everything will work as long as the handles themselves don't change.

Note. To get full credit it was *necessary* to point out that handles imply the OS can move memory around without the program's knowledge. Some people had a serious misconception about how handles work, thinking of them as a table of free blocks rather than a double-indirection to a single contiguous block; this is *incorrect*, but if the answers to parts (b) and (c) were at least consistent, partial credit was usually given. Note that it clearly states in the question that "a handle is a double indirection to a block of memory", so there is really *no* justification for treating a handle as naming an array of noncontiguous blocks making up a single allocation request.

c) [4] Describe *two* performance impacts that arise when programmers routinely use handles. (**Hint:** one occurs frequently, the other relatively infrequently.)

Answer. *Frequent:* every handle dereference is a double indirection, rather than a single indirection. This is significant since memory access is typically in the critical path of the performance of non-I/O-bound applications. *Infrequent:* when the memory allocator has to rearrange the heap and coalesce free blocks, a temporary stall (analogous to that introduced by garbage collection) could very likely occur. This problem is commonly observed in real systems that do implicit memory allocation. Partial credit was given for performance effects that, while they technically exist, are barely discernible compared to the above effects.

3 Filesystems and Leases [12]

You're designing an NFS-like network file system for Unix that allows many clients to access and edit files on a network-connected remote fileserver. We'll refer to the server as *S* and three clients as *X,Y,Z*. When a client opens a file, the server sends a copy of the *whole* file to the client. In addition:

1. Any number of clients may simultaneously open a file for *read only* access.

2. If X opens the file for writing, a lock is set on the server so that future clients can *only* open that file for reading. When X closes the file, the lock is released so that future clients may open the file for writing. The entire act of writing X's changes and releasing the lock is atomic with respect to the server. The new contents of the file are *not* automatically sent to clients that have the file open for reading.

a) [3] Suppose X obtains a write lock on a file, and then crashes. When X reboots itself, it "forgets" which file(s) it had write locks on. What is the effect of this failure on the other clients and on the server?

Answer. The server cannot release the write lock on the file, so the file is locked against writing "forever".

To remedy the problem of (a), you suggest that the server use *leases*. A lease gives X the right to access the file *for a limited amount of time*. (As before, at most one client can hold a write lock on the file.) When that time expires, if X still wants to use the file, it must ask the server to renew the lease, otherwise the server will unilaterally terminate the lease (and release the write lock, if the leaseholder had one).

b) [4] Explain how leases fix the problem in the scenario of part (a), and explain any new effects seen by X in that scenario after it reboots.

Answer. If X dies and reboots, and it forgets that it had a file locked for writing, after a while the file's lease expires. Since X doesn't know to renew it, after the lease expires the server revokes the lease and releases the write lock, and at that point others are free to write the file.

Any changes previously made by X are lost, however. (Some credit was lost for failing to mention this effect.)

c) [3] Suppose X is more careful: when it obtains a lease, it also records the fact that it is editing a particular file, and locally saves changes to that file as edits are in progress. X now crashes, reboots, and allows the user to recover the *local copy* of the file she was editing. Describe a scenario and the circumstances under which X might perceive a filesystem inconsistency.

Answer. X locks file; X dies and restarts, but by the time it has restarted, the lease has expired and another client (say Y) has locked the file for writing. X now has local changes to the file that are inconsistent with what Y has written. If X tries to reacquire the write lock, it will lose its changes (since it will get the new copy of the file from the server). The answer "X still thinks it has the lease" is incorrect: if X really recorded the fact that it had a lease, it can determine after rebooting if the lease period has expired since it last renewed. Even so, no inconsistency would actually be seen by X until the above scenario occurs.

d) [2] Assume that you can guarantee that any client's recovery time after a crash is at most R. Describe one possible way to avoid the inconsistency of part (c) , and describe its effect on the system.

Answer. If the current lease period is L, extend it to $L+R$. That is, if a lease renewal does not come, allow a "grace period" of length R, to account for the fact that X might have crashed and will renew the lease as soon as it recovers. The effect is that if X really doesn't care about the file (it didn't crash but doesn't care about renewing the lease), the other clients who are waiting for the write lock are forced to wait R longer than before. To get full credit you had to mention (at least in some general terms) the slowdown effect this causes on the system under normal operation.

4 Debugging Breakpoints [15]

You have been assigned the job of adding *data breakpoints* to an existing C debugger. The desired behavior is that the programmer can "mark" particular variables as being breakpoints: whenever a marked variable is *read or modified*, the debugger should take a breakpoint and allow the programmer to inspect the program's state, etc., then resume execution.

Assume that you are *not* allowed to make the programmer modify or recompile her source code, but you *do* have full access to the operating system and the runtime system, and in particular you can modify the virtual memory functions of the operating system (page fault handlers, page tables, etc.).

Also assume (as is the case in most implementations) that the compiler and linker arrange to store *global* variables in a designated memory pages that is known at link time, that all global variables will fit on one page, and that that page is not used for storing anything other than global variables.

a) [4] Suppose first that we only care about being able to mark global variables. Explain in detail how you would use the OS's virtual memory system to implement data breakpoints. In your explanation, keep in mind that only *some* global variables are likely to be marked.

Answer. Mark the pages containing global variables as inaccessible by the user process. This will cause an OS trap (page fault) on every access to the page containing global variables. Use the faulting address (supplied by the page fault handler) to determine whether the faulting access was to a marked variable (very important); if so, transfer control to the debugger, otherwise resume the user program.

Some answers were written in a manner that assumes *every* memory access can be individually checked by a piece of code to determine if a breakpoint has been hit. This is impossible; the virtual memory system does not consist of arbitrary code that can be executed on every access. Marking pages invalid traps accesses and can *then* cause the inspection code to check if a breakpoint has been hit.

b) [4] Qualitatively describe the impact on the overall speed of execution when the programmer marks a variable. What factor(s) dominate this impact? To what extent does the *number* of marked variables influence performance (assuming all marked variables are referenced equally often)?

Answer. Every global variable reference, whether to a marked variable or not, will result in a page fault and a trip through the breakpoint logic. The dominant factor is the very high cost (in most systems) of taking a software interrupt: user-to-kernel crossing, context switch to new address space at higher privilege, etc., and the switch back to user space to resume the program. The number of variables does not matter (as long as they are contained on a single *page*) since any access to that page will fault. There may be a cost associated with determining *which* breakpoint was hit, and that cost *does* vary with the number of marked variables; but that cost is miniscule compared to the cost of handling the page fault. There is no associated disk access cost, since global-variable page faults will *not* generally cause disk access.

c) [3] Suppose we instead want to support only *modify* breakpoints: a marked variable should cause a breakpoint only when its value is *modified*, not when it is read. What modifications could you make to your implementation to support modify-breakpoints more efficiently than read-breakpoints?

Answer. Mark the appropriate pages read-only (rather than invalid/disallowed). Faults occur only when the page is modified, not accessed.

d) [4] Describe what additional complication(s) you would encounter if you also had to implement this feature for functions' local variables. Identify *at least one* important factor that would affect performance if this feature is added, above and beyond the performance effects already discussed.

Answer. Because local variables are allocated on the stack, their addresses are not known until the function declaring those variables is called. Detecting when this occurs would require some kind of support for setting code breakpoints at function entry points. The two deleterious effects on performance would be: (a) even if we can detect when a function is called and when it returns, we must now modify the page tables at each of those events. This is costly because it requires a change in privilege level or user-space-to-kernel-space crossing. (b) Assuming local variables tend to be referenced more frequently than global variables, which is usually the case, we would be trapping many more memory accesses compared with only marking globals.