Comprehensive Exam: Algorithms and Concrete Mathematics Autumn 2001

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 an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.*

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

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

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

_____

(Number)

Prob	# 1	# 2	# 3	# 4	# 5	Total
Score						
Max	10	15	10	10	15	60

1. (*10 points*) *Big-Oh notation, Running times*
For each of the following functions, circle all of the correct answers. Note that any single mistake (an incorrect answer marked or one of the correct answers not marked) will cost you all of the points for this question. No need to provide proofs.

   (a) (*3 points*) $f(n) = n^{\epsilon}, 0 < \epsilon \leq 1, \epsilon$ constant.
   Choose: $O(n)$ $O(n^2)$ $O(n^3)$ $O(\log n)$ $O(n \log n)$ $O(2^n)$ none

   (b) (*3 points*) $f(n) = n^{\frac{\log \log n}{\log n}}$
   Choose: $o(\log n)$ $O(\log n)$ $\omega(\log n)$ $\Omega(\log n)$ $O(n)$ $O(n \log n)$ none

   (c) (*4 points*) $f(n) = \log^*(\log n)$
   Choose: $o(\log^*(\log^2 n))$ $\Theta(\log^* n)$ $O(\log^* n)$ $o(\log \sqrt{n})$ none

2. (*15 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) (*3 points*) $T(n) = T(\sqrt{n}) + n, T(2) = 1.$

   (b) (*4 points*) $T(n) = 5T(n/4) + \sqrt{n}, T(2) = 1.$

   (c) (*4 points*) $T(n) = 0.5(n^2)T(n/2), T(2) = 1.$

   (d) (*4 points*) $T(n) = 2T(\frac{n}{3} + \log n) + n, T(1000) = 1.$
   ("1000" was chosen to emphasise that you should not be concerned with small values of n.)

3. (*10 points*) You might recall that, in certain types of weighted graphs, Dijkstra's shortest path algorithm does not produce a correct shortest path. Present an example where Dijkstra's algorithm will not produce a correct answer even though a shortest path exists and it is unique. In other words, the algorithm will produce a wrong value of a distance between two points. Explain step-by-step the execution of the algorithm on your example and point out why it fails.

4. (*10 points*) Consider the following experiment: you have n balls and n bins. Balls are numbered 1 through n. First you put each ball into a bin chosen uniformly at random from 1 to n. Let X_i denote the number of balls in the ith bin. Prove that expectation of $\max_i X_i$ is bounded by $O(\log n)$.

5. (*15 points*) You are given n items with associated weights $w_1, w_2, \ldots, w_n$ and costs $c_1, c_2, \ldots, c_n$. The costs are all integers drawn from range $[0 \ldots 1000]$. The goal is to choose a subset of items such that the total weight will not exceed a given value W while the total cost will be maximized. Explain a polynomial time algorithm to solve this problem. Prove that your algorithm is correct and compute its asymptotic running time.

   Is your algorithm still polynomial if instead of $[0 \ldots 1000]$, the range is $[0 \ldots k]$, where k is part of the input? Explain.

2001 Comprehensive Examination
Artificial Intelligence

1. Search. (20 points) Consider a search tree with uniform branching factor b and depth d, and consider a search problem for which there is a single solution in the tree at depth k. A solution at the root of the tree is depth 0. Give expressions for the worst case cost of finding the solution, in terms of nodes visited, for (a) breadth-first search, (b) depth-first search, and (c) iterative deepening (starting at depth 0 and incrementing by 1 on each iteration. Give closed form expressions, if you can; but sums are okay. If you are unable to do this problem in general, you can still get some points by answering the question for the special case of b=2. And, if that is still too daunting, you may be able to scrape out a point or two by fixing k and d as well.

2. Automated Reasoning. (30 points) Two questions related to resolution.

(a) Consider the following pairs of expressions. u, v, w, x, y, z are variables; all other letters are constants. For each pair, say whether or not they are unifiable; if the answer is yes, give the most general unifier.

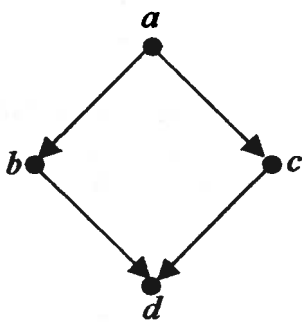
$p(x,b)$ and $p(f(y,y),y)$
$q(x,f(y,a),g(g(x)))$ and $q(z,f(z,u),y)$

(b) Given the following premises, use the resolution method to prove $\neg p(c,a)$.

$\forall y.\ \forall z.(p(y,z) \Rightarrow \neg p(z,y))$
$\forall x.(p(b,x) \Rightarrow p(a,x))$
$p(b,c) \vee p(a,c)$

Note that this is a question about the resolution method. You will get zero points for proving it in any other way.

3. Probability. (30 points) Adapted from Nilsson's *Artificial Intelligence: A New Synthesis*. The admissions committee for a major university wants to know the probability that an applicant is qualified given that the person is admitted. It has the belief network shown below.



6

$p(a)$=0.5
$p(b|a)$=1
$p(b|\neg a)$=0.5
$p(c|a)$=1
$p(c|\neg a)$=0.5
$p(d|b,c)$=1
$p(d|b, \neg c)$=0.5
$p(d|\neg b,c)$=0.5
$p(d|\neg b, \neg c)$=0

a - applicant is qualified
b - applicant has a high grade point average
c - applicant has a high graduate record examination score
d - applicant is admitted

What is the probability that an admitted student is qualified? In other words, calculate $p(a|d)$.

4. Natural Language. (20 points) Consider the augmented phrase structure grammar shown below.

$S(r(x, z) \wedge r(y, z)) \rightarrow Q(r(both(x,y), z))$
$Q(w(u, v)) \rightarrow NP(u)\ Verb(w)\ NP(v)$
$NP(x) \rightarrow Noun(x)$
$NP(both(x, y)) \rightarrow NP(x)$ ***and*** $NP(y)$
$Noun(tom) \rightarrow$ ***Tom***
$Noun(dick) \rightarrow$ ***Dick***
$Noun(harry) \rightarrow$ ***Harry***
$Noun(mary) \rightarrow$ ***Mary***
$Verb(hates) \rightarrow$ ***hate***
$Verb(hates) \rightarrow$ ***hates***

(a) Given that s is the top-level non-terminal, is there a semantic interpretation for the expression ***Mary hates Tom and Harry***? If so, what is it?

(b) Given that s is the top-level non-terminal, is there a semantic interpretation for the expression ***Tom and Harry hate Mary***? If so, what is it?

(c) Change the augmentations on the existing rules to eliminate ungrammatical sentences like *Tom and Harry hates Mary* (without eliminating the corresponding grammatical sentences). If you are unable to do this, you can still get partial credit by changing the rules themselves.

SOLUTIONS

2001 Comprehensive Examination
Artificial Intelligence

1. Search. If we assume that the choices at each branch point are searched left to right, then the worst case occurs when the solution lies at depth k down the rightmost branch of the tree.

In breadth-first search, the tree is searched only to the level at which the solution lies. This leads to the following expression.

$$\frac{b^{k+1}-1}{b-1}$$

In depth-first search, each branch is explored to its greatest depth before backing up. We search the entire tree except for the nodes lying below the solution. This leads to the following expression.

$$\frac{b^{d+1}}{b-1}-\frac{b^{d-k+1}}{b-1}+1$$

Each iteration in iterative deepening search repeats the work of the previous iteration. The following expression gives the worst-case cost.

$$\sum_{i=0}^{k}\frac{b^{i+1}-1}{b-1}$$

The good news is that, like breadth-first search, it does not need to search to greater depth than the solution. Also, since the tree is growing exponentially, this repeated work is always bounded by the amount of work in searching the next level; so the increase in cost over the other methods is not very great. In the case of a uniform tree, iterative deepening searches no more that $b/(b\text{-}1)$ worse than breadth-first search in terms of nodes searched.

2. Automated Reasoning.

(a) $\{x \leftarrow f(b,b), y \leftarrow b)\}$

(b) Not unifiable. Applying the unification algorithm up to the last argument of each expression, we get the substitution $\{x\leftarrow z, y\leftarrow z, u\leftarrow a\}$ and the clauses $h(z,f(z,a),g(g(z))$, and $h(z,f(z,a),z)$. At this point, the occurs check prevents us from binding z to $g(g(z))$.

(b) In resolution we first add the negated goal to the premises. In this case, the goal $\neg p(c,a)$ becomes $p(c,a)$ after negation. We then convert the premises and negated goal to clausal form, leading to the following clauses.

$\{\neg p(y,z), \neg p(z,y)\}$
$\{\neg p(b,x), p(a,x)\}$
$\{p(b,c), p(a,c)\}$
$\{p(c,a)\}$

We then apply the resolution rule to these clauses to produce the empty clause.

1.	$\{\neg p(y,z), \neg p(z,y)\}$	Premise
2.	$\{\neg p(b,x), p(a,x)\}$	Premise
3.	$\{p(b,c), p(a,c)\}$	Premise
4.	$\{p(c,a)\}$	Negated Goal
5.	$\{p(a,c)\}$	2,3
6.	$\{\neg p(c,a)\}$	1,5
7.	$\{\}$	4,6

3. Probability. One way to solve the problem is shown below.

$$p(a \mid d) = \frac{p(a,d)}{p(d)}$$

$$p(a,d) = p(a,b,c,d) + p(a,b,\neg c,d) + p(a,\neg b,c,d) + p(a,\neg b,\neg c,d)$$

$$p(\neg a,d) = p(\neg a,b,c,d) + p(\neg a,b,\neg c,d) + p(\neg a,\neg b,c,d) + p(\neg a,\neg b,\neg c,d)$$

$$p(d) = p(a,d) + p(\neg a,d)$$

Using the given probabilities, we can compute the following values.

$p(a,b,c,d) = 0.5$
$p(a,b,\neg c,d) = 0$
$p(a,\neg b,c,d) = 0$
$p(a,\neg b,\neg c,d) = 0$
$p(\neg a,b,c,d) = 0.125$
$p(\neg a,b,\neg c,d) = 0.0675$
$p(\neg a,\neg b,c,d) = 0.0675$
$p(\neg a,\neg b,\neg c,d) = 0$

Using these values and the equations above, we get the following results.

$$p(a,d) = 0.5$$

$$p(d) = 0.75$$

$$p(a \mid d) = 2/3$$

Note that one can take a short cut in this case by noticing that $p(d|a)=1$, i.e. all qualified applicants are admitted (though some non-qualified applicants are also admitted). From this observation, one can directly compute $p(a,d)=0.5$ using the definition of conditional probability.

4. Natural Language.

(a) There is none.

(b) *hates(tom,mary)* & *hates(harry,mary)*

(c) One way is to add a number parameter, as shown below.

S(r(x, z) ∧ r(y, z)) → Q(r(both(x,y), z))
Q(w(u, v)) → NP(u,n) Verb(w,n) NP(v)
NP(x,s) → Noun(x)
NP(both(x, y),p) → NP(x) ***and*** *NP(y)*
Noun(tom) → ***Tom***
Noun(dick) → ***Dick***
Noun(harry) → ***Harry***
Noun(mary) → ***Mary***
Verb(hates,p) → ***hate***
Verb(hates,s) → ***hates***

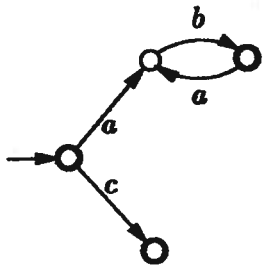
It is also possible to accomplish this by splitting the rules for *qs*, *np*, and *verb*; but this can be more cumbersome.

10

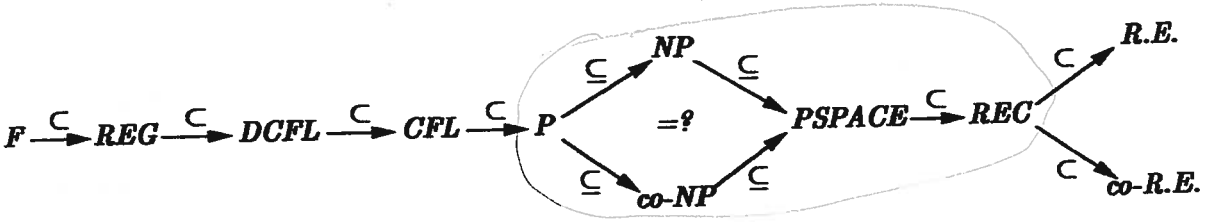
Solutions comprehensive exam Automata and Formal Languages

October 31, 2001

1. *Draw a nondeterministic finite automaton without ε-moves accepting the language* $(\mathbf{ab})^* + \mathbf{c}$.



2.



3. *Consider the following problem: Given descriptions of two Turing machines M and N over a given alphabet, is there a string accepted by both M and N?*
   - *Is this problem decidable?*

   No. By Rice's theorem it is undecidable whether a given Turing machine M accepts a non-empty language. That problem can be reduced to the problem above, by taking N to be a fixed Turing machine that accepts every string over the given alphabet. It follows that also the problem above is undecidable.
   - *Is it recursive enumerable?*

   Yes. Given descriptions of M and N, enumerate all pairs (w, n) with w a word over the given alphabet and n a natural number. For each such pair (w, n) run both M and N on w for n steps. In case both accept, return "Yes".

4. *A* clique of size k *in an undirected graph G is a subgraph with k nodes, wherein every two nodes are connected by an edge. In particular, there must be an edge between each node in the clique and itself (a* self-loop*). Let a* soft clique *be obtained by skipping the requirement that there are self-loops. Thus, a soft clique of size k is a subgraph with k nodes, wherein every two different nodes are connected by an edge. Now every clique is surely a soft clique.*

CLIQUE is the problem that asks, given an undirected graph G and a number k, whether G has a clique of size k. Likewise, let SOFT CLIQUE be the problem that asks, given an undirected graph G and a number k, whether G has a soft clique of size k.

Given the fact that the problem CLIQUE is NP-complete, prove that SOFT CLIQUE is NP-complete as well. Be concise.

The NP-hardness of SOFT CLIQUE can be established by means of a polynomial time reduction from CLIQUE to SOFT CLIQUE. Given an instance of CLIQUE, i.e. an undirected graph G and a number k, create the subgraph $G^\bullet$ of G consisting of all nodes in G with self-loops and all edges of G between such nodes. As any clique in G must lay entirely within $G^\bullet$, and in $G^\bullet$ any soft clique is a clique (and vice versa), G has a clique of size k if and only of $G^\bullet$ has a soft clique of size k. It remains to be shown that the reduction (i.e. the creation of $G^\bullet$) can be done in polynomial time. The following algorithm establishes that: pass over the description of G (a list of nodes and edges) once, listing all nodes with self-loops. Then pass over the description again, this time listing all edges between such nodes. This is clearly a polynomial time algorithm.

Besides showing NP-hardness, it should also be pointed out that SOFT CLIQUE is in the class NP. One way to do that is by means of a polynomial time reduction from SOFT CLIQUE to CLIQUE. Given an instance of SOFT CLIQUE, i.e. an undirected graph G and a number k, create the graph G° out of G by adding a self-loop for every node. This is clearly a polynomial time reduction. G has a soft clique of size k if and only if G° has a clique of size k.

5. *Prove that the language $L = \{xyy \mid x, y \in \{0,1,2\}^*,\ |y| > 0\}$ is not context-free.*
Context-free languages are closed under intersection with a regular language. Thus, if L is context-free then also $K := L \cap 2(0+1)^*22(0+1)^*2 = \{2y22y2 \mid y \in \{0,1\}^*\}$ is context-free. However, the latter is not context-free, as can be established with the pumping lemma.

Suppose K is context-free. Let p be the pumping length for this language.
Then $z := 20^p1^p220^p1^p2 \in L$. So it must be possible to write z as $uvwxy$, such that

(a) for each $i > 0$, $uv^iwx^iy \in L$,

(b) $|vx| > 0$, and

(c) $|vwx| \leq p$.

In case v or x contain a 2, pumping down (by taking $i = 0$) yields a string that is not in K. In case v and x both lay in the left half of z, pumping down disturbes the matching between the left and right sides of the string. The same holds if v and x both lay in the right half of z. In case v lays to the left, and x to the right of the middle 2's, by condition (c) above, v contains only 1's and x only 0's. Again, pumping down disturbes the matching between the left and right sides of the string. It follows that K, and hence also L, is not context-free.

Directly using the pumping lemma to show that L is not context-free is rather hard.

Compilers Comprehensive Exam Solutions (Fall 2001)

This is a 30 minute, closed book exam. The questions (except for the last) are based on the attached context-free grammar for a toy programming language. The grammar is intended for an LALR parser generator, such as YACC or Bison. It has some "semantic actions" but many have been omitted.

1. When this grammar is run through an LALR parser generator (e.g., YACC) 18 shift-reduce conflicts are reported.

   (a) Most of the conflicts involved the operators +, -, *, and /. What is the problem, and how can it be fixed easily without changing the grammar rules?
   This is a simple problem. The grammar is ambiguous because the relative precedence of the arithmetic operators is not specified. The grammar could be re-written in this case, but the problem says not to change it. So, the solution is to use the feature of YACC to specify precedence using declarations such as `%left`.

   (b) There are two conflicts that seem qualitatively different from the previous ones. One is a shift/reduce conflict when the next input is the token `ID`, and the two items in the offending state are:

   ```
   block     ->  BEGIN typedecls . constdecls vardecls stmts END   (rule 2)
   typedecls ->  typedecls . typedecl   (rule 3)
   ```

   There is a similar problem with `constdecls` and `vardecls`. Explain what is going on here and suggest a way to fix it. Be specific.
   The problem has to do with lookahead. The parser needs to know whether it is at the end of the list of typedecls to determine whether it should reduce the empty production for constdecls or not. But the lookahead symbol in either case is same: "ID." If the next construct is a constdecl or vardecl, it should reduce the `constdecl -> /* empty */` *production. Otherwise, it should shift the ID (part of the right hand side of the typedecl production).*
   This cannot be solved by tweaking precedence, since the parser genuinely needs to shift some times and reduce others, based on insufficient information. My inclination would be to change the language to allow declarations to occur in arbitrary order. This eliminates the need to decide when the typedecls end, simplifies the grammar, and might make life easier for the programmer. There are many other language changes that could address the problem as well.
   However, we don't always have the freedom to change the language. We could generalize the language anyway, build a syntax tree, and write code to check that everything was in the right order in the tree.

2. Suppose you wanted to write a straightforward recursive descent parser for the same language (assume the parser does not use backtracking, and does not look ahead more than one token). What problems would the context-free grammar pose in its current form? First, would the conflicts in the previous question cause problems? Second, would there be additional problems? Explain your answers.

   Recursive descent parsing is based on LL(1) parsing. All of the above would still be problems, because LALR(1) parsing almost deals with a superset of the grammars LL(1) parsing can deal with. Ambiguity is definitely a problem.

An additional major problem is left recursion in the grammar. The grammar would have to be re-written to eliminate it.

Finally, the various declarations all begin with the same tokens, which will prevent it from being LL(1). This can be solved by adding a new non-terminal to represent the ~~comment~~ common prefixes of the right-hand sides of these productions.

3. Describe what the `arithtype(t1, t2)` function should do to implement C-like type semantics for arithmetic operators, including error detection.

   `t1` and `t2` should be pointers to records representing data types. Both types should be INTEGER or FLOAT, otherwise, it should report an error. The return type should be INTEGER if both are integers or FLOAT if one or both are FLOATS.

4. Almost all machines have different instructions for integer and floating-point operations, and instructions to convert between them. Suppose you were implementing a simple code generator for this language. When would an instruction to convert an integer value to floating point value be emitted?

   If one argument to, say, + is INTEGER and the other is FLOAT, an instruction should be generated to convert the integer argument to a float before using a floating point add instruction to compute the value of the expression.

5. Describe what `checkassign` function should do, addressing the following questions:

   (a) What information should the arguments contain?

   (b) What errors should be reported? (It is important to list only those errors that would be detected here, as opposed to other productions in the grammar.)

   The arguments should be data structures describing variable declarations (whatever is built in the declarevar function call). We need to check that the lhs is declared to be a variable, not a types or constants (it's illegal to assign to constants). Then we need to check that the type of the assigned expression is compatible with the type that was declared for the lhs variable.

6. Are there circumstances under which performing the optimization of "common subexpression elimination" would slow down the compiled program? Explain your answer.

   Common subexpression elimination saves computation by storing and reusing subexpressions that appear more than once in the program. It is undesirable if the cost of storing the value is greater than the cost of computing it. This might be true because the cost of computing it is incredibly cheap (e.g., incrementing a variable), or because fast storage locations (registers) are heavily used for other purposes, such as loop index variables, at a particular part of the program.

```
%token BEGIN END TYPE CONST INTEGER FLOAT RECORD VAR ASSIGN ID INT_CONST FLOAT_CONST

%start program

%%

program         :       block

block           :       BEGIN typedecls constdecls vardecls stmts END
                ;

typedecls       :       typedecls typedecl
                |       /* empty */
                ;

typedecl        :       ID ':' TYPE type ';' { declaretype($1, $4); }
                ;

type            :       INTEGER { $$ = inttype(); }
                |       FLOAT { $$ = floattype(); }
                ;

vardecls        :       vardecls vardecl
                |       /* empty */
                ;

vardecl         :       ID ':' VAR type ';' { declarevar($1, $4); }
                ;

constdecls      :       constdecls constdecl
                |       /* empty */
                ;

constdecl       :       ID ':' CONST type const ';' { declarevar($1, $4); }
                ;

const           :       INT_CONST
                |       FLOAT_CONST

stmts           :       stmts ';' stmt
                |       /* empty */
                ;

stmt            :       block
                |       lhs ASSIGN expr ';' { checkassign($1, $3); }
                ;
```

```
lhs             :       ID { $$ = checkvardecl(lookupdecl($1)); }
                ;

expr            :       lhs { $$ = vartype($1); }
                |       const { $$ = consttype($1); }
                |       expr '+' expr { $$ = arithtype($1, $3); }
                |       expr '-' expr { $$ = arithtype($1, $3); }
                |       expr '*' expr { $$ = arithtype($1, $3); }
                |       expr '/' expr { $$ = arithtype($1, $3); }
                |      '(' expr ')' { $$ = $2; }
                ;

%%
```

num: type Integer
VAR integer

Computer Science Comprehensive Examination Solution
Computer Architecture

Autumn 2001

(Total time = 60 minutes, Total Points = 60)

Magic Number:_____Solution Set_____

In recognition of and in the spirit of the Stanford University Honor Code, I certify that I will neither give nor receive unpermitted aid on this exam.

Magic Number Signature:_____

This examination is open book. Please do all your work on these sheets. Do not do your work in a blue book. You have one hour to complete the exam. Before starting, please check to make sure that you have all 10 pages.

1	10	
2	8	
3	6	
4	8	
5	10	
6	6	
7	12	
Total	60	

1. Short Answer Question (10 points)

Assume that you have written the following function:

```
double AddCond(double *a, double *b, double startVal)
{
   if (a != NULL) {
        startVal = startVal + (*a);
        if (b != NULL) {
             startVal = startVal + (*b);
        }
   }
   return startVal;
}
```

After you turn in the function to your professor you re-read the instructions and discover that the function could not include any branch instructions! You decide your best hope is to submit an addendum to your assignment that describes a compiler and architectural features that allow this function to be generated with no branch instruction.

Describe here the architectural features you are assuming the compiler has to work with. Show the branch-less assembly language of this function in a pseudo assembly language of your choosing.

Let's assume we have some kind of predicated or conditional instructions.
Assume that a is passed in Ra, b is passed in Rb, and startVal is passed in Fv.

```
SetNe R1, Ra,0          // Set R1 with (a != NULL)
SetNe  R2,Rb,0          // Set R1 with (b != NULL)
SetAnd  R3,R1,R2        // Set R3 with (a != NULL) && (b != NULL)
CondLoad  F1,(Ra),R1    // if (R1) then load F1 with *a
CondLoad  F2, (Rb),R3   // if (R3) then load F2 with *b
CondAdd  Fv,F1,R1       // if (R1) then add F1 to Fv
CondAdd  Fv,F2,R3       // if (R3) then add F2 to Fv
Ret Fv                  // return startVal;
```

More space for question 1 (if needed)

2. Short Answer Question (8 points total)

Assume you are given two caches: A and B. Cache A is an 64 byte direct-mapped cache. Cache B is an 64 byte 4-way set associative cache with LRU replacement. Both caches use a 16 byte line size. Assume that you have a memory reference address stream that is listed in column one of the table below. Fill in the table using the following notion:

Confl - The cache will take a conflict miss.
Capac - The cache will take a capacity miss.
Compul - The cache will take a compulsory miss.
Hit - The cache will hit.

Reference Address	Cache A	Cache B
0x0000	***Compul***	***Compul***
0x0148	***Compul***	***Compul***
0x028c	***Compul***	***Compul***
0x03c0	***Compul***	***Compul***
0x0004	***Conflic***	***Hit***
0x0408	***Compul***	***Compul***
0x014c	***Capac***	***Capac***
0x000c	***Confl***	***Hit***

3. Short Answer Question (6 points)

Although numerous studies of cache architectures have shown limited miss-rate benefits and high implementation cost of having a very high degree of associativity in primary data caches, successful machines have been constructed with 32way and higher associativity in their data cache. Explain why an implementation might use a high degree of associativity that isn't justified by cache miss rate reduction.

You do this when you want to increase the cache capacity without increasing the number of bits used as an index. This is most frequently done so the cache can be index using the virtual memory page offset bits allowing the TLB lookup to be done in parallel with cache access.

4. Short Answer Question (8 points)

Traditionally the TLB of a CPU has not been as transparent to the OS as other caches such as the instruction and data caches. This lack of transparency means that implementations that add a TLB frequently extend the architecture to make the TLB visible to the software and require the operating system be aware of this cache. Describe how an implementation could transparently add an effective TLB to existing paged architecture without requiring OS changes.

The TLB need to maintain consistency with the in-memory page tables of the virtual memory system. Having the TLB snoop on changes made to memory and detect if the cached PTE is being modified. This is similar to an invalidation-based multiprocessor cache coherency protocol.

5. Short Answer Question (10 points)

The design of the MIPS and SPARC RISC architectures were done when a simple 5-stage pipeline was a popular implementation technique for their first implementation. Although both designs support register+offset addressing, the MIPS choose not to implement register+register addressing except for loads and stores accessing the registers of the floating-point unit.

(a) Describe how the initial pipeline design would have influenced the MIPS architects to make this decision on addressing modes.

(b) In comparison between the MIPS and SPARC architecture, the SPARC architects claim that better CPI seen for the MIPS architecture was caused in part by the existence of register+register addressing in SPARC and not in MIPS. Explain this.

(A) Having register+register on a store instruction with a 5-stage pipeline can require 3 registers (value being stored and two address register) fetch at once. No other instruction needs 3 read ports into the register file to fetch its operands in a single cycle. Note that the supporting register+register to the FP register still only requires two ports in the integer register file.

(B) ***Without register+register the MIPS architecture requires separate add instructions to do the add while the SPARC can do the add with the store instruction that stalls for one additional cycle. Since MIPS runs more instructions and those instructions take only a single cycle it results in a lower CPI that SPARC can executes fewer but instructions with more stalls.***

6. Short Answer Question (6 points)

Explain how scatter/gather I/O interfaces to high bandwidth I/O devices such as disks is important to support large bulk data transfers in modern computer systems.

Modern computer systems use virtual memory so that large data blocks are likely to be discontinuous in physical memory. Scatter/gather DMA is used to put together these pages for transfer to and from the I/O device.

7. True and False Questions (3 points each, 12 points total)

Answer the following questions as either True or False and provide a brief justification of your answer.

(a) An architecture with condition codes (e.g. a flags register) in which every instruction sets every condition code will show no instruction level parallelism on a dynamic scheduled super-scalar architecture because of the dependencies on the condition code register. (True or False, justify)

False, a dynamic scheduled architecture can rename the flags register like any other register allowing ILP.

(b) A VLIW architecture will have better instruction density than a standard 32-bit RISC instruction set. (True or False, justify)

False, VLIW architectures frequently have worse density because of the need to pad with NOPs when ILP is lacking.

(c) It is always possible to build a program that will defeat any branch predictor so that the branch prediction rate will be less than 10%. (True or False, justify)
False, a random branch predictor could not be defeated in this way.

(d) WAW hazards are only possible if instructions take a variable number of cycles to execute. (True or False, justify)
True, fixed number of cycles mean the instructions finish in order so WAW is not possible.

Stanford University Computer Science Department
2001 Comprehensive Exam in Databases

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

MAGIC NUMBER: _____

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

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.

Stanford University Computer Science Department
2001 Comprehensive Exam in Databases
SAMPLE SOLUTIONS

1. **Entity-Relationship Model** (8 points)

An office "coffee club" assigns three tasks each day; the tasks are: (1) Clean the pot. (2) Brew the coffee. (3) Buy the supplies. Each day, each of the jobs is assigned to one of the members, and no member can be assigned to two jobs on the same day. We wish to design a database in which we shall store the assignments of jobs to members for each day, and also store some information about members: their name, phone number and email address. You may assume there are more than three members in the club. The database should enforce the constraints on assignments of jobs as best it can. Give an appropriate E/R diagram for this database, and explain your reasoning.

2. **Functional Dependencies** (7 points)

Suppose we are given a relation $R(A, B, C, D)$ with functional dependencies $ABC \rightarrow D$ and $D \rightarrow AB$.

(a) What are the keys for R?
$\{A, B, C\}$ *and* $\{C, D\}$

(b) Is R in third normal form? Explain your reasoning.
Yes; all attributes are prime, so there cannot be a violation.

(c) Is R in Boyce-Codd normal form? Explain your reasoning.
No; D is not a superkey, so $D \rightarrow AB$ is a BCNF violation.

3. **Multivalued Dependencies** (7 points)

Consider a relation $R(A, B, C)$, and suppose that in each of the columns A, B, and C, no value can appear more than once (although the same value could appear in two or three different columns).

(a) List all of the nontrivial multivalued dependencies satisfied by R.

$A \twoheadrightarrow B, A \twoheadrightarrow C, B \twoheadrightarrow A, B \twoheadrightarrow C, C \twoheadrightarrow A, C \twoheadrightarrow B$

(b) Suppose the constraint on R is relaxed so that column A can have a value appearing more than once, but B and C still do not have duplicate values in their columns. Now, list all of the nontrivial MVD's satisfied by R.

$B \twoheadrightarrow A, B \twoheadrightarrow C, C \twoheadrightarrow A, C \twoheadrightarrow B$

4. **Relational Algebra** (10 points)

Suppose we are given a relation *Sells*(*bar, beer, price*), whose tuples (a, b, c) mean that bar a sells beer b at price c. Write in the "classical" relational algebra (operations union, intersection, difference, selection, projection, product, renaming, and the natural and theta-joins) the following queries. You may use complete expressions, expression trees, or sequences of assignments to local variables as you wish.

(a) Find the bars that sell two different beers at the same price.

```
Sells1(bar,beer1,price) := Sells
Temp1(bar,beer,beer1,price) := Sells NATURAL JOIN Sells1
Temp2(bar,beer,beer1,price) := SIGMA_{beer!=beer1}(Temp1)
Answer(bar) := PI_bar(Temp2)
```

(b) Find the greatest price at which any beer is sold at any bar.

```
Temp1(price) := PI_price(Sells)
Temp2(price1,price2) := Temp1 * Temp1
Temp3(price1,price2) := SIGMA_{price1<price2}(Temp2)
Temp4(price) := PI_price1(Temp3)
Answer(price) := Temp1 - Temp4
```

5. **SQL** (12 points)

A school chess team maintains rankings of its players based on scores, using the following relation:

```
Player(name, score) // name is a key
```

(a) For the first part of the problem assume that scores are unique within the relation. Write a single SQL query over the `Player` relation that takes a player name as an argument (denoted `:n` in the query) and returns the player's ranking by score. For example, if the `<name, score>` tuples in the `Player` relation are:

```
<Tim, 168>
<Kelly, 130>
<Shelby, 129>
<Miles, 110>
<Emma, 92>
<Rachel, 75>
```

then for `:n = Tim` the query result should be 1, for `:n = Shelby` the query result should be 3, for `:n = Rachel` the query result should be 6, etc. Write the query here:

```
SELECT COUNT(*)
FROM   Player
WHERE  score >=
  (SELECT score FROM Player P where P.name = :n)
```

(b) Now suppose that scores are not unique within relation `Player`. In this case, the players are in groups that are ranked according to score. You are to write a single SQL query that takes a player name as an argument (denoted `:n` in the query) and returns the rank of the player's group. For example, if the `<name, score>` tuples in the `Player` relation are:

```
<Tim, 168>
<Kelly, 130>
<Shelby, 130>
<Miles, 130>
<Emma, 92>
<Rachel, 75>
<Richard, 75>
```

then for `:n = Tim` the query result should be 1; for `:n = Kelly`, `:n = Shelby`, or `:n = Miles`, the query result should be 2; for `:n = Emma` the query result should be 3; for `:n = Rachel` or `:n = Richard` the query result should be 4. *Remember that simplicity of solutions does count.* Write the query here:

```
SELECT COUNT(DISTINCT score)
FROM   Player
WHERE  score >=
  (SELECT score FROM Player P where P.name = :n)
```


30

6. **Constraints** (4 points)

What familiar constraint type is encoded by the following SQL general assertion? *Your answer should contain at most two words.*

```
CREATE ASSERTION Mystery AS
   (NOT EXISTS (SELECT * FROM R
                WHERE R.A NOT IN (SELECT B FROM S)))
```

Answer: *referential integrity*

7. **ODL and OQL** (12 points)

Consider the following ODL (Object Definition Language) schema.

```
interface Applicant (extent Apps, key SSN) {
  attribute integer SSN;
  attribute Struct<string first, string last> name;
  relationship Set<Job> applied
     inverse Job::applicants; }

interface Job (extent Jobs, key (company, city) {
  attribute string company;
  attribute string city;
  relationship Set<Applicant> applicants
     inverse Applicant::applied }
```

(a) Is the relationship between applicants and jobs enforced by this schema one-one, one-many, many-one, or many-many?

many-many

(b) What is the simplest modification we can make to the schema to enforce that each applicant can apply for only one job?

Change `relationship Set<Job> applied`
to `relationship Job applied`.

(c) Using OQL (Object Query Language), write a query to find the SSN and last name of all applicants who have applied for a job in Palo Alto. Do not repeat (*SSN,last-name*) pairs in the result, even if the applicant has applied for many jobs in Palo Alto.

```
SELECT DISTINCT Struct(a.SSN, a.name.last)
FROM   Apps a, a.applied j
WHERE  j.city = "Palo Alto"
```

Comprehensive Exam: Graphics
Autumn 2001

This 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 onto the back of the page if necessary.

You have 60 minutes to complete the exam.

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)

1. [Total: 20 points] Color and intensity.

1A [5 points]. A CRT converts voltage to light intensity. What is the form of the curve used to convert voltage to intensity? If you double the voltage, approximately how much light will be produced?

1B [5 points]. When light enters your eye it creates the sensation of brightness. Suppose you double the amount of light energy entering you eye from a point in the scene, approximately how much will the sensation of brightness change?

1C [5 points]. Are the three color phosphor colors (the so-called R, G and B phosphors) sufficient to generate any color perceivable color? Why or why not?

1D [5 points]. Why do printers use cyan, magenta and yellow inks as their primary colors? In addition, most printing processes use a fourth ink black. Why is using a fourth ink useful?

2. [Total: 20 points] Transformations.

2A [10 points]. Linear transformations may be represented as matrices. An example of a linear transformation is a rotation. A rotation is a special kind of matrix called an orthogonal matrix. An orthogonal matrix has the property that the inverse of the matrix is equal to its transpose. Prove that rotation matrices are orthogonal. Are there other types of matrices that are orthogonal?

2B [10 points]. All non-degenerate transformations (i.e. all except those that map multiple points to the same point) have an associated inverse transformation. Why are inverse transformations useful in computer graphics? Suppose you are given a sequence of transformations, A then B then C then D, what is the inverse of this sequence of transformations?

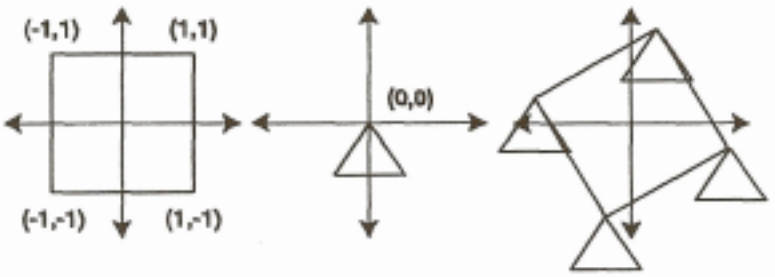
3. [Total: 20 points] Hidden surface elimination.

3A [5 points] Describe the z-buffer algorithm for hidden surface elimination.

3B [15 points] Give three advantages of the z-buffer algorithm over other means for doing hidden surface elimination. Give two disadvantages.

4. [Total: 20 points] Hierarchical modeling.

As part of your new job at Optical Arts, you've been asked to build a simple game with a Ferris wheel. This Ferris wheel is very simple, consisting of a square and a diamond-shaped car for passengers. Assume you have been given two procedures, one to draw a unit square (`square()`)and another to draw a triangular car (`triangle()`). The figures drawn by these procedures are shown below.



Write a graphics program that draws the Ferris wheel at any angle of rotation. Assume the graphics library has transformation commands (the syntax is not important; i.e. make up your own syntax). Make sure that the triangular car is positioned so that its center of gravity is below the point of attachment.

5. [Total: 20 points] Rasterization.

You are building a new graphics library, and you receive a request to write a procedure to draw the following exponential function

$$y^2 = 2^x - 1$$

To simplify the problem, consider only positive y values. This procedure should draw n steps along the curve starting at the origin. The procedure should ensure that each point is connected (that is, is adjacent) to the previous point. The procedure should also be written as efficiently as possible. That is, use the minimal number of arithmetic operations in the inner loop. To draw the curve, write down an implicit function that defines the set of points on the curve, and then incrementally trace out the curve. Work out the math, and then fill in the following template:

```
drawexponential(int n)
{
  int x = 0;
  int y = 0;




  for(i=0; i<n; i++) {
    point(x,y);









  }
}
```

Comprehensive Examinations in Logic
November 2001

30 questions

Time: 1 hour

Instructions

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

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

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

WRITE MAGIC NUMBER: _____

38

Section A

For each question in this section you need to choose one out the four possible choices provided. If you answer correctly you get **two points**. Answering incorrectly will result in **two points** being **deducted** from your score. Indicate correct answer by writing your choice clearly in the box provided. **No points deducted for leaving questions unanswered.**

1. A sentence ψ is valid **if and only if**

   (a) $(\neg\ \psi)$ is satisfiable.

   (b) ψ is satisfiable.

   (c) $(\neg\ \psi)$ is valid.

   (d) $(\neg\ \psi)$ is unsatisfiable

2. Using deductive tableaux for propositional logic with resolution and the **"polarity strategy"**. Which of the following statements is correct:

   (a) Given any propositional sentence ψ, we can prove ψ valid.

   (b) Given any propositional sentence ψ, we can prove ψ satisfiable.

   (c) If ψ is valid, we can find a proof using resolution and the polarity strategy.

   (d) Given an invalid sentence (ψ), we can prove $(\neg\ \psi)$ valid.

3. Which of the following statements is necessarily true about a well-founded relation $<$ over $A \times A$?

   (a) A is an infinite set.

   (b) $<$ is irreflexive.

   (c) $<$ is transitive.

   (d) A is a finite set.

4. Let Σ be a **set** of first-order sentences. Which of the following statements about Σ is necessarily true?

   (a) If $\Sigma \models \psi$ and $\Sigma \models \neg\psi$ for some ψ then every interpretation is a model of Σ.

   (b) If Σ is unsatisfiable then every subset of Σ is unsatisfiable.

   (c) If $\Sigma \models \psi$ and $\Sigma \models \neg\psi$ for some ψ then Σ does not have a model.

   (d) If Σ is empty then there is no ψ such that $\Sigma \models \psi$.

5. Let $\mathcal{F}$ be a valid sentence in predicate logic.

   I $\mathcal{F}$ is also valid in the theory of natural numbers **with** 0,1,addition and multiplication.

   II $\mathcal{F}$ is valid in the theory of natural numbers **with** 0,1, addition and **without** multiplication.

   III There are **first–order** theories $\mathcal{T}$ such that $\mathcal{T} \not\models \mathcal{F}$.

   Which of the statements above are necessarily true?

   (a) II only

   (b) I and II only

   (c) I,II and III

   (d) none of them.

6. Let $\mathcal{F}$ and $\mathcal{G}$ be sentences in propositional logic and let $\mathcal{H}$ be the sentence (*if* $\mathcal{F}$ *then* $\mathcal{G}$). If $\mathcal{G}$ is not satisfiable then which of the following is necessarily true?

   (a) $\mathcal{H}$ is valid.

   (b) If $\mathcal{H}$ is not valid then $\mathcal{F}$ is valid.

   (c) The sentence ($\mathcal{H}$ *or* $\mathcal{F}$) is valid.

   (d) None of the above.

40

7. Let $\mathcal{F}$ and $\mathcal{G}$ be sentences in first-order logic. Given that $\mathcal{F}$ is valid precisely when $\mathcal{G}$ is valid, which of the following statements are necessarily true?

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

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

(a) I only.

(b) II only.

(c) Both I and II.

(d) neither I nor II.

8. Given a sentence $\mathcal{F}$, we obtain $\mathcal{G}$ by removing all its quantifiers through **validity preserving** skolemization. Which of the following statements is true about $\mathcal{F}$ and $\mathcal{G}$?

(a) $\mathcal{G}$ is equivalent to $\mathcal{F}$.

(b) $\mathcal{G}$ may not exist for any given $\mathcal{F}$.

(c) It is always possible to skolemize in such a way that $\mathcal{G}$ is equivalent to $\mathcal{F}$.

(d) If $\mathcal{F}$ and $\mathcal{G}$ are not equivalent then $\mathcal{F}$ is not valid.

9. You are given a first–order sentence ψ such that ψ has an infinite model. Which of the following are **necessarily** true

I ψ has a countably infinite model.

II ψ has a model of any infinite cardinality.

III There is some infinite cardinality κ_0 such that ψ does not have a model of cardinality κ_0.

(a) I and II only.

(b) I and III only.

(c) II only.

(d) I only.

10. Which of the following statements is necessarily true about a set Σ of first-order sentences?

    (a) Consistency of Σ is decidable.

    (b) No finite subset of Σ is consistent.

    (c) If $\Sigma \models \psi$ then there is some finite subset Γ of Σ s.t. $\Gamma \models \psi$.

    (d) If $\Sigma \models \psi$, every subset of Σ entails ψ.

Section B

For each question in this section you need to write "yes" if you think the statement holds or "no" if you think it does not. You receive **two** points if you answer correctly. However **one** point will be deducted for wrong answers. There is no penalty for leaving questions unanswered.

1. If a sentence φ is satisfiable and does not have a finite model, $\neg \varphi$ is satisfiable.

2. If a sentence φ is not satisfiable in any finite model but has an infinite model, $\neg \varphi$ has a countably infinite model.

3. Any finitely axiomatizable first-order theory is recursive.

4. The transitive closure of a well-founded relation is itself well-founded.

5. We start with the empty set and for every first-order sentence ψ that is not equivalent to any sentence previously added to the theory, we add either ψ or $(\neg \psi)$. Any theory so formed is consistent.

6. The following terms are unifiable

$$h(x,\ f(y,a),\ y) \quad \text{and} \quad h(f(f(b,\ z),\ z),\ x,\ f(z,a))$$

7. The result of removing all the quantifiers of the sentence $\mathcal{F}$ below by validity preserving skolemization is equivalent to $\mathcal{F}$.

$$\mathcal{F}: \quad \textit{if} \ (\forall y)(\exists z)p(y,z) \ \ \textit{then} \ \ (\forall z')(\exists y')p(y',z')$$

8. Given a relation $R \subseteq A \times B$, we define its symmetric closure as the smallest symmetric relation R' containing R. It is possible for the symmetric closure of some well-founded relation over a non-empty domain to be well-founded.

9. In the deductive tableaux for propositional logic, given a sentence ψ, we write the sentence in the goal column as G_0. $G_{i+1},\ i \geq 0$ is obtained by applying $\mathcal{GG}$ resolving G_i with itself according to the polarity strategy. If ψ is valid, the procedure terminates with "`true`" as a final goal.

10. We can use the procedure above to prove a given sentence non-valid.

11. It is possible to write a first-order sentence that is false on all infinite models.

12. There is a finitely axiomatizable first-order theory whose only countably infinite model is the set of natural numbers, with 0, 1, addition, multiplication, less than relation.

13. If resolution is applied going against the "polarity strategy" one can prove a non-valid sentence.

14. The set of Gödel numbers of valid sentences in the language of Peano Arithmetic is recursive.

15. There are statements that can be proven using **Complete** Induction and not by **Step-wise** Induction.

16. $\{(\ if\ \ p\ \ then\ \ p_2), (\ if\ \ p\ \ then\ \ p_1),\ p\} \models (p_1\ \ or\ \ p_2)$

17. Any first-order theory with equality can be converted to a theory without equality that has precisely the same models as the original theory, by adding a finite number of sentences that axiomatize equality.

18. Given any two models, we say that they are elementarily equivalent if and only if they satisfy the same first-order sentences. Is elementary equivalence of two finite models decidable?

19. Given a first-order theory Σ, we define $\mathcal{M}(\Sigma)$ as the set of models of Σ. Similarly given any set of models Φ, we define $Th(\Phi)$ as the set of First order sentences that are true in all the models of Φ. If Σ is complete and consistent, then $Th(\mathcal{M}(\Sigma)) = \Sigma$.

20. The propositional implication connective is complete (i.e., *and*, *or*, $\neg$ may be derived from implication).

LOGIC COMP (cerrected solution

Logic-solutions

Section A

1d	6c
2c	7D
3b	8d
4c	9a
5b	10c

Section B

1yes	6 no	11yes	16 yes
2yes	7 no	12no	17 no
3no	8 yes	13no	18 yes
4yes	9 yes	14no	19 yes
5No	10 yes	15no	20 no

Computer Science Department

Stanford University

Comprehensive Examination in Networks

Fall 2001

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. **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 where reasonable.

Problem 1 (10 points)

a. Why do network designers often separate network functionality into different protocol "layers?"

b. What is the purpose of each of the following Internet protocol stack layers?

Application layer
Transport layer
Network layer
MAC/link layer
Physical layer

c. True or False? IPsec or some other security/encryption scheme when used at the network layer makes application-layer security protocols unnecessary.

Problem 2 (10 points)

What are the advantages and disadvantages of the following two error control mechanisms?

a. A sliding window of one ("stop and wait")

b. "Go back N"

Problem 3 (10 points)

One problem that can occur on telephone lines is echoes. When a signal gets to the destination, some of the energy may be bounced back, causing annoying acoustic echoes.

Echo suppressors are devices that detect human speech coming from one direction of the connection and suppress all signals going the other way. When one person stops talking and the other starts, the echo suppressor switches directions. *Echo cancellers* are circuits that analyze the echo, estimate how big it is, and subtract it from the signal delivered without the need for mechanical relays.

What are the advantages of the use of echo cancellers over echo suppressors?

Problem 4 (10 points)

To initiate a TCP connection, a 3-way handshake is used. The initiator sends a SYN packet, the receiver acknowledges the SYN and the initiator then ACKS that acknowledgement. What problems does this handshake address, with respect to possible loss, reordering, duplication, and delay of packets at the network layer?

Problem 5 (10 points)

This problem concerns circuit switching versus packet switching.

For each of the following network characteristics, write "C" if it applies to circuit-switched networks, "P" if it applies to packet-switched networks, or "B" if it applies to both.

a. uses available bandwidth efficiently

b. as network load increases, bandwidth for particular conversations decreases

c. as network load increases, new conversations may fail to be established

d. handles intermediate router failure gracefully

Problem 6 (10 points)

This question concerns UDP.

Which of the following features does UDP provide? (Answer "yes", "no" or "optional" for each one.)

a. guaranteed delivery

b. data integrity (checksum)

c. duplicate protection

d. multiplexing (source/destination ports)

e. flow control

f. reordering protection

Computer Science Department
Stanford University
Comprehensive Examination in Numerical Analysis
Fall 2001

Points for each question and its parts are specified
Books and notes are [redacted] permitted

---

1 (10 points) Consider a linear least-squares data-fitting problem:

$$\min_{x \in R^n} \|Ax - b\|_2 \qquad (1),$$

A is a real $m \times n$ matrix, $m \geq n$, of full column rank, $b \in R^m$.

a) Describe the normal equations method for solving (1) (3pts)
b) Describe the Householder transformations method for solving (1) (4pts)
c) Compare the two abovementioned methods in terms of accuracy (relative error in x) and computational cost (number of flops) (3pts)

---

2 (10 points) Consider the equation

$$y'(t) = \lambda y(t)$$
$$y(0) = 1, \qquad \lambda \in R, \quad \lambda < 0$$

a) (2pts)Give the exact analytical solution to the ODE and discuss the solution as $t \to +\infty$.
b) (3pts) Suppose we use the trapezoidal rule

$$y_{k+1} = y_k + \frac{\lambda y_k + \lambda y_{k+1}}{2} h,$$
$$y_0 = 1$$

Under what conditions will $y_k \to 0$ as $k \to \infty$?
c) (5pts) Now consider a general initial value problem $y' = f(t, y)$, $y(0) = 1$, and the trapezoidal rule for solving it:

$$y_{k+1} = y_k + \frac{f(t_k, y_k) + f(t_{k+1}, y_{k+1})}{2} h,$$
$$y_0 = 1$$

where f is sufficiently well-behaved. In particular it's Lipschitz-continuous in its 2nd argument with the constant L. Let $t_k = kh$. Obtain an estimate for

$$| \, y(t_k) - y_k \, |.$$

You can express the answer in terms of constants dependent on $y(t)$.

---

3 (10 points) Numerical quadrature: Suppose we want to obtain an estimate for the integral of a function over an interval:

$$I(f) = \int_a^b f(x)dx \approx \sum_{i=1}^n w_i f(x_i), \qquad x_i \in [a,b], \quad x_1 < x_2 < \ldots < x_n$$

a) (3pts) Derive a two-point quadrature rule ($x_1 = a$, $x_2 = b$, $n = 2$) exact for polynomials of degree ≤ 1, e.g. by fitting a polynomial to the data points and integrating.
b) (4pts) Give an error estimate for this rule assuming that f(x) is sufficiently smooth.

c) (3pts) To attain arbitrarily high accuracy in evaluating an integral we can subdivide the original interval into subintervals, apply a low-order quadrature rule in each subinterval, and sum the results. This is called a *composite* quadrature rule. For example, if the interval $[a,b]$ (we don't care if the ends belong to the interval or not) is partitioned into n subintervals $[x_{i+1}, x_i]$, $i = 1,\ldots,n$, $a = x_0 < x_1 < \ldots < x_n = b$, the composite trapezoid rule has the form

$$I(f) \approx \sum_{i=1}^{n} (w_1^{(i)} f(x_{i-1}) + w_2^{(i)} f(x_i))$$

Derive an error estimate for this rule. For simplicity take the subintervals to be of equal length.

Solutions

1 a) Denote $r = Ax - b$; $x = \arg\min\|r\|_2 = \arg\min r^T r = \arg\min(x^T A^T Ax - 2(A^T b)^T x + b^T b)$

Differentiating with respect to x and setting the result to zero, we get $2A^T Ax - 2A^T b = 0$, or

$$A^T Ax = A^T b \qquad (2)$$

A is of full column rank => $A^T A$ is nonsingular => (2) has a unique solution.

$A^T A$ is symmetric and positive definite => we can compute its Cholesky factorization

$$A^T A = LL^T,$$

where L - lower triangular. This allows us to reduce (2) to solving the triangular systems

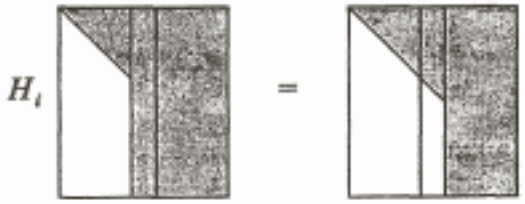
$$Ly = A^T b, \qquad L^T x = y.$$

b)First we want to decompose A into the product $Q\begin{bmatrix} R \\ 0 \end{bmatrix}$, $Q: m \times m$ orthogonal, $R: n \times n$ upper-triangular; since A is of full column rank, R is nonsingular.

To reduce A to the upper-triangular form, we successively apply Householder transformations

$$H_n ... H_1 A = \begin{bmatrix} R \\ 0 \end{bmatrix}, \qquad \text{where } H_i = I - 2\frac{v_i v_i^T}{v_i^T v_i} \quad \Rightarrow \quad \text{orthogonal and symmetric.}$$

H_i reflects a vector against the hyperplane $v_i^{\perp}$. We choose v_i so that H_i zeros out the subdiagonal part of the ith column $\tilde{a}_i$ of the current state of A - $(H_{i-1} ... H_1 A)$



If $\tilde{a}_i' = \{\tilde{a}_i$ with the upper i-1 entries set to 0$\}$, then $v_i = \tilde{a}_i' \pm \|\tilde{a}_i'\|_2 e_i$, where the sign is chosen so as to avoid cancellation.

Now, using this decomposition $Q^T A = \begin{bmatrix} R \\ 0 \end{bmatrix}$, $Q^T = H_n ... H_1$,

$$x = \arg\min\|Ax - b\|_2 = \arg\min\|Q^T Ax - Q^T b\|_2 = \arg\min\left\|\begin{bmatrix} R \\ 0 \end{bmatrix} x - Q^T b\right\|_2;$$

let $Q^T b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$, $b_1: n \times 1$, $b_2: (m-n) \times 1$;

$$x = \arg\min\left\|\begin{bmatrix} R \\ 0 \end{bmatrix} x - \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}\right\|_2^2 = \arg\min(\|Rx - b_1\|_2^2 + \|b_2\|_2^2) = \arg\min\|Rx - b_1\|_2 = R^{-1} b_1$$

c)

	Computations	Accuracy
Normal equations	A^TA - $\approx mn^2$ flops; Cholesky factorization - $\approx \frac{n^3}{3}$ flops Triangular systems - $O(n^2)$ $\approx mn^2 + \frac{n^3}{3}$ flops	relative error in x is proportional to $(cond(A))^2$
Householder transformations	$\approx 2mn^2 + \frac{2}{3}n^3$ flops	relative error in x is proportional to $cond(A) + \|r\|_2 (cond(A))^2$

For nearly square problems, $m \approx n$, the two methods require about the same amounts of work, but for $m >> n$ normal equations method is about 2 times cheaper than Householder method. On the other hand, the Householder method is more accurate.

2 a) Via elementary symbolic calculations we obtain $y(t) = e^{\lambda t}$. As $t \to \infty$, $y(t) \to +0$.

---

b) $y_{k+1}(1-\frac{\lambda h}{2}) = y_k(1+\frac{\lambda h}{2})$, $\quad y_{k+1} = \frac{1 + \lambda h / 2}{1 - \lambda h / 2} y_k$, $\quad y_k = \left(\frac{1+\lambda h/2}{1-\lambda h/2}\right)^k y_0$

$y_k \to 0 \quad \Leftrightarrow \quad \left|\frac{1+\lambda h/2}{1-\lambda h/2}\right| < 1, \quad (2+\lambda h)^2 < (2-\lambda h)^2, \quad 4\lambda h < -4\lambda h, \quad \lambda h < 0,$

and since we assume $h > 0$, this is true for all negative λ.

---

c) We have $\quad y_{k+1} = y_k + \frac{f(t_k, y_k) + f(t_{k+1}, y_{k+1})}{2} h \qquad (1)$

For the true solution $y(t)$ we can write

$$y(t_{k+1}) = y(t_k) + \frac{y'(t_k) + y'(t_{k+1})}{2} h + \xi, \qquad (2)$$

where ξ is some unknown term. From Taylor expansion we have two formulas:

$\frac{f(x+h) + f(x-h)}{2} = f(x) + \frac{\theta h^2}{2}, \qquad |\theta| \le \|f''\|_C;$

$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + \frac{\theta_1 h^2}{6}, \qquad |\theta_1| \le \|f^{(3)}\|_C.$

Regrouping (2) and applying these formulas, we get:

$\frac{\xi}{h} = \frac{y(t_{k+1}) - y(t_k)}{h} - \frac{y'(t_{k+1}) + y'(t_k)}{2} = y'(t_{k+1/2}) + \frac{\eta h^2}{6} - y'(t_{k+1/2}) - \frac{\eta_1 h^2}{2} = (\frac{\eta}{6} - \frac{\eta_1}{2})h^2$, where

$|\eta| \le \|y^{(3)}\|_C, \quad |\eta_1| \le \|y''\|_C$. Hence $\quad |\xi| \le \frac{1}{2}(\|y''\|_C + \|y^{(3)}\|_C)h^3. \qquad (3)$

Now denoting $\delta_k = y_k - y(t_k)$ and subtracting (1)-(2), we get

$$\delta_{k+1} = \delta_k + \frac{f(t_k, y_k) - f(t_k, y(t_k))}{2} h + \frac{f(t_{k+1}, y_{k+1}) - f(t_{k+1}, y(t_{k+1}))}{2} h - \xi.$$

Then using Lipschitz-continuity of f,

$|\delta_{k+1}| \le |\delta_k| + \frac{Lh}{2}|\delta_k| + \frac{Lh}{2}|\delta_{k+1}| + |\xi|, \qquad (1 - \frac{Lh}{2})|\delta_{k+1}| \le (1 + \frac{Lh}{2})|\delta_k| + |\xi|,$

$$|\delta_{k+1}| \le \frac{|1+Lh/2|}{|1-Lh/2|}|\delta_k| + \frac{|\xi|}{1-Lh/2} \le \frac{|1+Lh/2|}{|1-Lh/2|}|\delta_k| + 2|\xi| \quad \text{for} \quad h \le \frac{1}{L}.$$

To simplify this, consider the general situation: $x_{k+1} \le ax_k + b$. Applying this inequality recursively, we get $x_k \le a^k x_0 + b(a^{k-1} + a^{k-2} + \ldots + 1) = a^k x_0 + b\frac{a^k - 1}{a-1}$. The first term in our case disappears, because $y(0) = y_0 = 1$; therefore $|\delta_k| \le 2|\xi| \dfrac{\left|\frac{1+Lh/2}{1-Lh/2}\right|^k - 1}{\left|\frac{1+Lh/2}{1-Lh/2}\right| - 1}$.

Now since $k = t_k / h$, $\left|\frac{1+Lh/2}{1-Lh/2}\right|^k \xrightarrow[h\to 0]{} \frac{e^{\frac{Lt_k}{2}}}{e^{-\frac{Lt_k}{2}}} = e^{Lt_k}$, and hence

$$|\delta_k| \le 2|\xi| \frac{e^{Lt_k} + \varepsilon(h) - 1}{Lh}(1 - Lh/2) \le \frac{2|\xi|}{h} \frac{e^{Lt_k} + \varepsilon(h) - 1}{L},$$

where $\frac{2|\xi|}{h} \le (\|y''\|_C + \|y^{(3)}\|_C)h^2$, and $\varepsilon(h) \to 0$ as $h \to 0$. Thus $|\delta_k| \le Ch^2$ for sufficiently small h.

3 a) Let $l(x) = f(a) + (x-a)\frac{f(b)-f(a)}{b-a}$, then

$$I(f) = \int_a^b l(x)dx = f(a)(b-a) + \frac{(x-a)^2}{2}\Big|_a^b \frac{f(b)-f(a)}{b-a} = (b-a)(f(a) + \frac{f(b)-f(a)}{2}) = (b-a)\frac{f(a)+f(b)}{2}$$

---

b) Let $I_0 = \int_a^b f(x)dx$, $\quad \Delta = b - a$.

$$f(a+t) = f(a) + f'(a)t + f''(\xi)\frac{t^2}{2}, \quad I_0 = f(a)\Delta + f'(a)\frac{\Delta^2}{2} + A\frac{\Delta^3}{6}, \quad |A| \le \|f''\|_C$$

$$f(b-t) = f(b) - f'(b)t + f''(\xi_1)\frac{t^2}{2}, \quad I_0 = f(b)\Delta - f'(b)\frac{\Delta^2}{2} + B\frac{\Delta^3}{6}, \quad |B| \le \|f''\|_C.$$

Averaging the two expressions for I_0, we obtain

$$I_0 = \frac{f(a)+f(b)}{2}\Delta + \frac{A+B}{2}\frac{\Delta^3}{6} = I + C\frac{\Delta^3}{6}, \quad \text{where } C = \frac{A+B}{2}. \text{ This implies } |I_0 - I| \le \|f''\|_C \frac{\Delta^3}{6}.$$

---

c) Let $I_0^{(i)}$, $I^{(i)}$ be the true integral and our estimate correspondingly for the ith subinterval. Then

$$I_0 = \sum_{i=1}^n I_0^{(i)}, \quad I = \sum_{i=1}^n I^{(i)}.$$

$$|I_0 - I| \le \sum_{i=1}^n |I_0^{(i)} - I^{(i)}| \underset{(b)}{\le} \sum_{i=1}^n |C_i| \frac{\Delta_i^3}{6} \le \sum_{i=1}^n \|f''\|_C \frac{1}{6}\left(\frac{\Delta}{n}\right)^3 = \|f''\|_C \frac{\Delta^3}{6n^2} = \frac{\|f''\|_C (b-a)^3}{6n^2}.$$

Comprehensive Exam: Programming Languages Autumn 2001

This is a 30-minute closed-book exam and the point total for all questions is 30.

All of the intended answers may be written within the space provided. You may use the back of the preceding page for scratch work. If you to use the back side of a page to write part of your answer, be sure to mark your answer clearly.

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

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

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

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

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

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

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

_____

(Number)

Prob	# 1	# 2	# 3	# 3	# 3	Total
Score						
Max	4	4	4	8	10	30

1. (4 points) *Lifetime and Scope.* Define *lifetime of a data object* and *scope of a declaration.* Describe a situation in which two variable declarations with the same scope produce two locations with different lifetimes.

2. (4 points) *Static and Dynamic Scope.* Explain the difference between static and dynamic scope. Give an example of a language or language feature that uses dynamic scope.

3. (4 points) *Tail Recursion.* What is the primary advantage of optimizing tail recursive functions so that recursive calls are eliminated? Specifically, suppose that a call `f(n)` to an unoptimized tail-recursive function `f` runs in time and space $O(n)$. If an optimizing compiler performs the optimization commonly called "tail recursion elimination," what are the time and space requirements of a call `f(n)`? Use "big-oh" notation; your answers should be $O(1)$ or $O(n)$ or $O(n^2)$ or something of this form.

55

4. (8 points) *Parameter Passing*
The C programming language uses pass-by-value parameter passing, while C++ also has pass-by-reference.

   (a) (4 points) Describe the sequence of storage allocations and assignments associated with the execution of the function call `increment(y)` in the following C fragment.

```
void increment (int x) {
     x++;
};
int y=0;
...
increment(y);
```

   Explain why the call `increment(y)` does not increment `y`.

   (b) (1 points) Write the call to `increment`, passing `y`, so that the call to `increment` in the following fragment increments the value of `y`.

```
void increment (int * x) {
     (*x)++;
};
int y=0;
...
<call_to_increment>;
```


5b

(c) (3 points) Is the following C++ code, using pass-by-reference, more efficient than the pass-by-value code in part (a)? Explain. Why is pass-by-reference often more efficient?

```
void increment (int & x) {
     x++;
};
int y=0;
...
increment(y);
```

5. (*10 points*) *Type Conversion*
This question asks you to read two short passages from documents explaining C++ and comment on their content and the connections between them.

Explanation of type conversion, from "C++ in Hypertext":
Most operators assume that their operands are of the same type. Thus, it is not possible to add an integer and a string - what would be the meaning of such an operation? However, if this rule of 'sameness' were enforced too strictly, it would make C++ an awkward language. For example, it makes sense to add an integer to a long or to a double. To handle this C++ allows what are called implicit conversions (also called type coercions) of one operand to the type of a second operand - when the conversion is appropriate. In the example involving the addition of an int and a long, the integer would be converted to a long. In the example involving an int and a double, the integer would be converted to a double. Such conversions are called 'implicit' conversions because they take place automatically.

C++ provides a set of conversion rules (also called 'promotion' rules) that guide the type of implicit conversions that take place.

The built-in types of C++ are arranged in order from 'lower' to 'higher' types, where a value of a lower type can be implicitly converted to a value of a higher type. 'Higher' here essentially means that the type can hold all the information contained in a 'lower' type and no information is lost in a conversion. Thus, C++ will implicitly convert an int to a double but not a double to an int. In the latter case, information is lost when the 'data' to the right of the decimal point is truncated.

Here is part of the conversion hierarchy:

```
double
float
long
int
char
```

Explanation of type conversion, from "C++ FAQ Lite":
Is it OK to convert a pointer from a derived class to its base class?

Yes. An object of a derived class is a kind of the base class. Therefore the conversion from a derived class pointer to a base class pointer is perfectly safe, and happens all the time. For example, if I am pointing at a car, I am in fact pointing at a vehicle, so converting a Car* to a Vehicle* is perfectly safe and normal:

```
void f(Vehicle* v);
void g(Car* c) { f(c); }  // Perfectly safe; no cast
```

Questions

(a) (*1 points*) Based on the first passage, what do you believe are the relative sizes (number of bytes) of `double`, `float`, `long`, `int`, `char`.

(b) (*2 points*) Why does this passage "'Higher' here essentially means that the type can hold all the information contained in a 'lower' type and no information is lost in a conversion." suggest that there is a correlation between implicit conversion and the number of bytes used to represent a datum?

50

(c) (*2 points*) If `Car` is a derived class with public base class `Vehicle`, which class of object is likely to require a larger representation in memory?

(d) (*2 points*) Is the order of conversion from `Car *` to `Vehicle *` consistent with the order of conversation from `int` to `float`? Explain.

(e) (*3 points*) Does it make more sense to convert from `Car` to `Vehicle` or `Vehicle` to `Car`? Explain one problem with each form of conversion.

2001 Programming Languages Comp. Solutions

written by 2006 Ph.D. First-Years

1. *Lifetime and Scope:* Within a function, if one variable is declared `static` and the other one is declared non-static, then the two variables have the same scope (accessible only within that function) but have different lifetimes (the static variable has a lifetime lasting the duration of the program execution while the non-static variable has a lifetime lasting the duration of a particular function call).

2. *Static and Dynamic Scope:* Static scope is resolved at compile-time (use the value of the variable declared in the closest enclosing block); dynamic scope is resolved at execution-time (use the value of the variable closest to the current function's activation record on the stack). Exceptions are a language feature that uses dynamic scope.

3. *Tail Recursion:* The primary advantage of tail recursion is space savings. Time requirement is still $O(n)$ but space requirement is $O(1)$ because new activation records aren't being pushed onto the stack for each recursive call.

4. *Parameter Passing:*

   (a) At the beginning of the call to `increment()`, the value of `y` on the stack (0) is copied to the slot where `x` is allocated on the stack. Then `x` is incremented to 1, and when the function returns, the stack pointer is moved back up and the value of `x` is lost. The call doesn't increment `y` because its value was copied to another location, and the value in that location (not `y`) is incremented.

   (b) `increment(&y)`

   (c) The C++ pass-by-reference code is a bit less efficient than the pass-by-value C code in part (a) because it must perform a pointer dereference before incrementing the value of `x`. However, it actually does the correct thing, which is to increment the value of `y` in the calling function, whereas the pass-by-value code doesn't do anything useful. In general, pass-by-reference is more efficient because you can pass a pointer to a large data structure instead of passing the data structure itself by value, which requires that a copy be made on the stack.

5. *Type Conversion*

   (a) `sizeof(double)` $\geq$ `sizeof(float)` $\geq$ `sizeof(long)` $\geq$ `sizeof(int)` $\geq$ `sizeof(char)` (that's implied by the hierarchy in the passage, but we don't think it's true in general because a `long` is usually longer than a `float`)

   (b) A type higher in the hierarchy contains more bytes than a type lower in the hierarchy, so no information is lost when converting from lower to higher because all of those bytes can fit (usually with more room to spare).

   (c) `Car` probably requires a larger representation in memory because it contains at least all the fields of `Vehicle`, and most likely, some additional fields.

(d) No, it's not consistent, because converting from `Car*` to `Vehicle*` is converting a pointer referring to something with a larger representation in memory (subclass) into something with a smaller representation in memory (superclass), whereas converting from `int` to `float` is converting from something with a smaller representation to something with a larger representation.

(e) It makes more sense to convert from `Car` to `Vehicle`. A problem with converting from `Car` to `Vehicle` is that you can't access Car-specific fields. A problem with converting from `Vehicle` to `Car` is that you attempt to access Car-specific fields, which don't exist in `Vehicle`. The latter is a more serious problem because you can read junk data.

Stanford University Computer Science Department

Fall 2001 Comprehensive Exam in Software Systems

SOLUTIONS

1. **CLOSED BOOK**: no notes, textbook, computer, Internet access, etc.
2. **WRITE ONLY IN BLUE BOOKS: No credit** for answers written on these exam pages.
3. **WRITE MAGIC NUMBER** on the cover of **EACH** blue book.
4. The exam is designed to take less than an hour.
5. If you need to make assumptions to answer a question, *state them clearly*.
6. **For questions that refer generically to "the OS": if you think the answer would differ substantially depending on the OS in question, assume any popular flavor of Unix (Linux, BSD, Solaris, etc.).**

Short Answers (3 pts each, 30 pts total)

1) Explain how dynamically linked libraries (DLL's or *.so* libraries) facilitate in-place OS or application software upgrades. (Hint: you can do this by contrasting them with statically-linked libraries.)
One can replace (upgrade) a DLL without replacing or modifying the applications that use it. With static libraries, in order to take advantage of a new version of the library, each app that used it would have to be recompiled.

2) Why are DLL's now back in favor, after being out of favor during the early days of the PC revolution?
Libraries have gotten large and machines can run multiple apps at once; having multiple copies of a static library rather than sharing one copy among multiple apps quickly eats up memory.

3) When a context switch occurs and a user process must give up the CPU, what *hardware* state information is saved by the OS in order to be able to later resume that user process?
Program counter, stack pointer, integer register file, FP register file (if present), processor flags (arithmetic and possibly privilege level).

4) This information is not sufficient to capture the state of the whole process from the OS's point of view. Name two other types of state that must be captured, and specify where they are kept.

Various choices of things kept in the process control block, including: page table base register

for the process; open file descriptors or other I/O endpoints (eg sockets), ie references to kernel data structures about open files. Page tables themselves are also acceptable.

5) Name one advantage of user-level threads over kernel-level threads.

   No kernel crossing to do a thread switch, so thread switches within a process are lower overhead.

6) Name one advantage of kernel-level threads over user-level threads.

   Scheduling can be done at finer grain (all user-level threads for a process compete for the process's scheduled time and resources).

7) What is the difference between authentication and authorization?

   Authentication proves you are who you say you are. Authorization says you have the right to do something.

8) Asynchronous I/O devices can be handled by either polling or interrupts. Describe one advantage of using polling over interrupts.

   Interrupts incur high processing overhead (because of the kernel crossing) and require hardware support.

9) Describe one advantage of using interrupts over polling.

   Interrupts occur exactly when an interesting event happens; polling can miss important events (if it's not frequent enough) or waste time checking too frequently (if important events are rare).

10) Name one difference between paging and segmentation.

    Page sizes are fixed to one (or a very few) particular sizes; segments can usually be of essentially arbitrary length.

Slightly Longer Answers (5 pts each, 30 pts total)

11) You look up the documentation for a particular Unix library function that you want to use. According to the documentation, the library function's implementation is reentrant. What does this mean, and under what circumstances would you care about this fact?

    It means multiple loci of control (typically, multiple threads) can be in the function at the same time. You would care if the function is being called in a multithreaded program, because you need to know whether to enforce mutual exclusion for threads that might call the function.

12) To successfully prevent user programs from causing damage to other programs or the OS, hardware support is required. Name two hardware support mechanisms in modern CPU's that enable this.

13) For each mechanism, briefly describe how it is used by the OS and in what specific way(s) it protects the OS or other user programs (i.e. what specific kind(s) of damage are prevented).

Virtual memory/page tables: allows OS to give each process its own address space and protects them from each other. OS also runs in its own address space, protecting it from user programs.

Privilege levels: some operations can only be done in "high privilege" or supervisor mode. This allows OS the exclusive ability to manage I/O and other resources, rather than exposing them to abuse by user programs.

14) A RAID system can fail if two or more of its drives crash within a short time interval. Suppose each drive fails once an hour with probability p, and that drive failures are independent. What is the probability of a *k*-drive RAID system failing in a given hour?

Probability that zero drives fail during a given hour: $(1-p)^k$. Probability that exactly one drive fails: $kp(1-p)^{k-1}$. Therefore, probability that 2 or more fail: $1-(1-p)^k-kp(1-p)^{k-1}$.

15) You want to transmit a large text file securely to a friend over a slow network. You decide to use both compression and encryption on the file. Which should you apply first and why?

Compress first, because the redundancy in text will be eliminated when the text is encrypted, so that the compression step would have nothing to do.

16) Ethernet and other CSMA networks use *exponential backoff* when a transmit collision is detected. Briefly explain what this is, and concretely explain the intuition behind using exponential (as opposed to some other function) backoff.

The more senders there are, the more likely are collisions. If everyone starts with a random backoff, even if they are staggered, they're more likely to collide again if conditions are crowded. With exponential backoff, the relative staggering of the re-sends quickly gets very far apart, so the probability of *repeated* collisions does not grow linearly with the number of senders.