CS109A Final Examination

Wednesday, March 20, 1996, 12:15–3:15PM

Directions

The exam is *open book*; any written materials may be used. Answer all 16 questions on the exam paper itself. The total number of points is 300.

If you do not wish to do heavy arithmetic, you may leave answers as a product of integers or as a quotient of products of integers. You **may not** leave answers in terms of factorials, permutations (Π), combinations $\binom{n}{m}$, or other formulas. Also, you must **cancel common factors** from numerator and denominator. Calculators are **permitted**.

You may make additional comments at the bottoms of pages or at the end if you feel the need to clarify or justify your answers.

Do not forget to **sign the pledge** below.

I acknowledge and accept the honor code.

____

Print your name here: ____

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

Problem 1: (20 points) For each of the following logical expressions, tell whether it is (A) a tautology (i.e., true for every truth assignment) or (B) satisfiable (i.e., true for at least one truth assignment) but not a tautology, or (C) not satisfiable. Answer A, B, or C.

a) $q + p\bar{q}$. ________

b) $(p + q)(\bar{p})(\bar{q})$ ________

c) $(p \rightarrow q) \rightarrow (q \rightarrow p)$ ________

d) $(p \rightarrow q) + \bar{p}$ ________

Problem 2: (15 points) For each of the following pairs of functions, tell whether the first is big-oh of the second. If not, just say "no." If the first *is* big-oh of the second, say "yes" and give **minimal** witnesses n_0 and c. By "minimal," we mean that lowering either n_0 or c while keeping the other fixed results in a pair that is *not* a witness. Note that there are usually many different choices of minimal witnesses when the big-oh relationship holds.

a) Is n^2 $O\big((n/2)^2\big)$?

b) Is $(\log_2 n)^3$ $O(n)$?

c) Is $2n^3$ $O(n^2 + n^4)$?

Problem 3: (15 points) The following ML function **sum** computes the sum of the elements on an integer list.

```
fun sum([x:int]) = x
|   sum(x::xs) = x + sum(xs);
```

Prove that fact by induction. Do not forget that you are proving something about the particular function sum, not a general statement about sums.

a Give the statement $S(n)$ to be proved. Be sure to make clear what n is.

b) What is an appropriate basis case? ____________

c) Prove the basis case.

d) Prove the inductive step $S(n) \rightarrow S(n+1)$.

Problem 4: (20 points) Let $S(n,m)$ be the set of sets of m integers chosen from $\{1,2,\ldots,n\}$. For example, $S(3,2) = \{\{1,2\},\{1,3\},\{2,3\}\}$. Note that $0 \le m \le n$ may be assumed. We can define $S(n,m)$ inductively as follows:†

BASIS: For any $n \ge 0$, $S(n,0)$ is $\{\emptyset\}$, that is, the set containing only the empty set of integers. For any $m > 0$, $S(0,m) = \emptyset$ (not $\{\emptyset\}$!).

INDUCTION: For $m > 0$ and any $n \ge m$, $S(n,m)$ is the union of $S(n-1,m)$ and the set $T(n-1,m-1)$ that is formed by inserting integer n into each set in $S(n-1,m-1)$. For example, note that $S(3,1) = \{\{1\},\{2\},\{3\}\}$. Thus, $T(3,1) = \{\{1,4\},\{2,4\},\{3,4\}\}$. Consequently, $S(4,2)$ is the union of this set $T(3,1)$ and the set $S(3,2)$ mentioned above.

Write an ML function `sets(n,m)` that computes $S(n,m)$. You should represent sets by lists, as usual. Assume there is available to you a function `appendAll(x,L)` that takes a list of lists L and appends to each list on the list L the element x. For example, `appendAll(3,[[1],[1,2]])` produces the list `[[1,3],[1,2,3]]`.

____

____

____

____

____

____

Problem 5: (20 points) Consider the recurrence

BASIS: $T(1) = 1$.

INDUCTION: $T(n) = 3T(n-1) + 2^n$ for integers $n > 1$.

Note that since there are no symbolic constants, this is not a "big-oh" problem; we can solve for $T(n)$ exactly.

Also note that the answers below should be simplified. For example, replace arithmetic or geometric sums by their equivalents (formulas are on p. 42 FCS, Exercises 2.3.9 and 2.3.10), and combine terms in algebraic expressions whenever you can.

a) To begin, what are the initial values of $T(n)$?

$T(1) =$ ______; $T(2) =$ ______; $T(3) =$ ______; $T(4) =$ ______;

† You may think of the induction as being a complete induction on the product mn.

b) Expand the inductive rule so $T(n)$ is expressed in terms of $T(n-2)$.

---

c) Express $T(n)$ in terms of $T(n-3)$.

---

d) What is the general pattern? That is, express $T(n)$ in terms of $T(n-i)$.

---

e) For what value of i can we eliminate $T(n-i)$ from the expression? $i =$________

f) Use your answer to (e) to express $T(n)$ as a function of n alone (i.e., without any terms involving the function T). You may get a series to sum, but you should be able to apply the formula on p. 42, FCS to get this sum in closed form.

$T(n) =$________________.

Problem 6: (15 points) Let $H(n)$ be the number of ways we can place the integers 1 through n into an array $A[1]$ through $A[n]$ so the array forms a heap. That is, $A[i] \leq A[i/2]$ for all $i = 2, 3, \ldots, n$. Give the first six values of H: $H(1) =$________; $H(2) =$________; $H(3) =$________; $H(4) =$________; $H(5) =$________; $H(6) =$________.

Problem 7: (15 points) Consider the probability space consisting of the fifteen points corresponding to the integers 1 through 15. Let E be the event "the integer is divisible by 2," and let F be the event "the integer is divisible by 3."

a) What is the probability of E? ________

b) What is the probability of F? ________

c) What is the probability of E/F? ________

d) What is the probability of F/E? ________

e) Is E independent of F in this probability space? ________

Problem 8: (20 points) At the Dandy Diner, there are very large supplies of red, white, and blue plates. There are also five seats at the counter, which we may think of as seats 1 through 5, numbered left to right. For each of the questions below, give the answer and, for possible partial credit, show your formula or reasoning briefly.

a) In how many ways can we put plates at each of the five counter seats? That is, a "way" to put plates is a sequence of five choices, one for each of the five seats, in order from the left; each choice is from one of the three colors of plates: red, white, and blue.

---

b) In how many ways can we put plates at each of the five counter seats if exactly two of the seats get a blue plate?

---

c) In how many ways can we put plates at each of the five counter seats so no two adjacent seats have the same color plate?

---

d) In how many ways can we select a distribution of the colors of the five plates, where two selections of plates are "the same" if they have the same number of white plates, the same number of blue plates, and the same number of red plates? Note that order of the plates at the counter is irrelevant for this part.

Problem 9: (30 points) Suppose we play "3-card poker," where hands consist of a set of three cards chosen from the ordinary 52-card deck. Tell how many cards fall into each of the following types. For partial credit, briefly show your formula or reasoning.

a) A pair; that is, two cards are of the same rank, and the other card is of a different rank.

---

b) Three of a kind; that is, all of the same rank.

---

c) A straight; that is, three cards of consecutive ranks. The order of the ranks is $2, 3, \ldots, 9, 10, J, Q, K, A$. Note that 2 does *not* follow A in this order.

---

d) A flush; that is, three cards of the same suit.

---

e) A straight-flush; that is, three cards of the same suit and of consecutive ranks.

---

f) Nothing; that is, any set of three cards that meets none of the descriptions (a) through (e).

Problem 10: (10 points) Here is a function `filter` in Curried form. Its first argument P is a predicate (i.e., a function that returns a `bool`), and its second argument is a list L. It returns the list of those elements of L that satisfy P.

```
fun filter P nil = nil
|   filter P (x::xs) =
        if P(x) then x::(filter P xs)
        else filter P xs;
```

Show how to apply `filter` to an anonymous function, such that the result is a function f; the function f takes a list of integers and returns the list of those elements that are less than 10. Your answer should be a single `val`-statement that assigns this function to the variable f.

Problem 11: (25 points) The following is a datatype representing unlabeled binary trees.

```
datatype btree =
    Empty |
    Node of btree * btree;
```

Values of type btree with *the same structure* can be defined recursively as follows:

BASIS: Two empty trees have the same structure. No empty tree has the same structure as a nonempty tree.

INDUCTION: Two nonempty trees have the same structure if and only if the left subtrees of the roots have the same structure and the right subtrees of the roots have the same structure.

a) Write an ML function `same(x,y)` that returns `true` if `btree`'s x and y have the same structure and `false` otherwise.

---

b) We can add labels to binary trees with a datatype:

```
datatype 'label lbt =
    Empty |
    Node of 'label * ('label lbt) * ('label lbt);
```

Write a function `samel(x,y)` that tells whether two binary trees x and y with integer labels are "the same," that is, they have the same structure and corresponding nodes have the same label.

Problem 12: (20 points) Below is a C function that prints the elements of a linked list of integers, represented by the usual element-next cells. It uses a while-loop.

```
void printList(LIST L) {
    while(L != NULL) {
        printf("%d\n", L->element);
        L = L->next;
    }
}
```

Write an ML function `printList(L: int list ref)` that mimics this C function as best you can, using a while-do-statement. Note that the argument L for your ML function is a reference to a list of integers; you do not have to create the reference with a `val`-statement.

Problem 13: (15 points) Write a function `inv(n,A)` that takes an integer $n \geq 1$ and an array of integers A and determines how many different values of i between 1 and $n-1$, inclusive, there are there that "invert" the order of elements, i.e., $A[i] < A[i-1]$.† To receive full credit, your function must be recursive and not use while-loops.

Problem 14: (20 points) A *full binary tree of height n* is defined recursively as follows.

BASIS: A single node is a full binary tree of height 0.

INDUCTION: If $n \geq 0$, then a full binary tree of height $n+1$ consists of a root node whose left and right subtrees are full binary trees of height n.

We wish to prove the following statement by induction on n:

$S(n)$: the sum of the heights of the nodes in a full binary tree of height n is $2^{n+1}-n-2$.

a) Prove the basis, $n = 0$, here.

b) Write the statement $S(n+1)$ here.

c) What is the height of the root in the full binary tree of height $n+1$? ______

d) Prove the inductive step, that $S(n)$ implies $S(n+1)$.

† Note this condition has nothing to do with the partially-ordered-tree property.

Problem 15: (20 points) Any two distinct nodes x and y in a tree are known to be related in exactly one of the following four ways: (A) x is a proper ancestor of y; (B) x is a proper descendant of y; (C) x is to the left of y; (D) x is to the right of y. When we list the nodes of the tree in preorder or postorder, we can get various relationships involving the order in which x and y appear in these two orders. For each condition below, indicate which of the four relationships A through D may hold?

a) x precedes y in both preorder and postorder. ______________

b) x precedes y in preorder but follows y in postorder. ______________

c) x precedes y in postorder but follows y in preorder. ______________

d) x follows y in both preorder and postorder. ______________

Problem 16: (20 points) There are a number of data structures besides a heap that we could use to implement the ADT PRIORITY QUEUE.

I. A linked list, with the elements sorted, *largest* first (i.e., in the reverse of the usual smallest-first order).

II. A binary search tree.

For each of these structures, suppose that n is the number of elements currently in the priority queue. In the table below, give simple, tight worst-case and average-case big-oh bounds on the running time of the two priority-queue operations *insert* and *deleteMax*. For the **average** case of the binary search tree, assume that the tree is approximately balanced, i.e., the maximum depth of a leaf is not too much greater (say twice) than the depth of leaves in full binary trees (as defined in Problem 14). For both the tree and the linked list structure, assume that in the **average** case the inserted element is equally likely to belong at any position in the tree or list. These assumptions do not apply for worst-case analysis, of course.

Data Structure	Worst-Case *insert*	Average *insert*	Worst-Case *deleteMax*	Average *deleteMax*
Sorted Linked List				
Binary Search Tree				