CS 70 Discrete Mathematics for CS
Spring 2005 Clancy/Wagner

MT 1

PRINT your name: ______________________________, ______________________________
(last) (first)

SIGN your name: ____

PRINT your class account name: `cs70-` _____________

Name of the person sitting to your left: ___________________________

Name of the person sitting to your right: ___________________________

You may consult any books, notes, or other paper-based inanimate objects available to you. Calculators and computers are not permitted. Please write your answers in the spaces provided in the test; in particular, we will not grade anything on the back of an exam page unless we are clearly told on the front of the page to look there.

You have 80 minutes. There are 5 questions, of varying credit (40 points total). The questions are of varying difficulty, so avoid spending too long on any one question.

Do not turn this page until your instructor tells you to do so.

Problem 1	
Problem 2	
Problem 3	
Problem 4	
Problem 5	
Total	

Problem 1. [True or false] (12 points)

Circle TRUE or FALSE. You do not need to justify your answers on this problem.

Reminder: $\mathbf{N} = \{0, 1, 2, 3, \dots\}$ represents the set of non-negative integers.

(a) TRUE or FALSE: The propositions $P \implies Q$ and $Q \implies P$ are logically equivalent.

(b) TRUE or FALSE: The proposition $Q \vee P \vee (P \implies Q)$ is guaranteed to be true, no matter what the truth values of P and Q may be.

(c) TRUE or FALSE: If we are told that the proposition P is true, we are entitled to conclude that $((P \implies Q) \implies P) \implies Q$ must also be true.

(d) TRUE or FALSE: $\forall n \in \mathbf{N} \,.\, (x^2 - x = 0) \vee (x \geq 2)$.

(e) TRUE or FALSE: $\forall x \in \mathbf{N} \,.\, \exists y \in \mathbf{N} \,.\, \forall z \in \mathbf{N} \,.\, x + y = z$.

(f) TRUE or FALSE: $\forall x \in \mathbf{N} \,.\, \exists y \in \mathbf{N} \,.\, x^2 = y$.

(g) TRUE or FALSE: $(\forall x \in \mathbf{N} \,.\, \exists y \in \mathbf{N} \,.\, x < y) \equiv (\exists y \in \mathbf{N} \,.\, \forall x \in \mathbf{N} \,.\, x < y)$.

(h) TRUE or FALSE: Let $P(n)$ denote the proposition that $1 + 2 + \cdots + n + (n+1) = n(n+1)/2$. Then $\forall n \in \mathbf{N} \,.\, P(n) \implies P(n+1)$ is true.

(i) TRUE or FALSE: Define the relation $\prec$ so that $n \prec m$ iff n divides m and $n < m$. Then the relation $\prec$ is a well-ordering on $\mathbf{N}$.

(j) TRUE or FALSE: Any 2-party cake-cutting protocol that is fair is also envy-free.

(k) TRUE or FALSE: In the stable marriage problem, if M denotes the male-optimal matching, then there exists a girl who does not get her optimal boy in M (i.e., her mate in M is not her optimal boy).

(l) TRUE or FALSE: Suppose that we have n boys, $b_1, \dots, b_n$, and n girls, $g_1, \dots, g_n$. Assume that g_1 lists b_1 as her top choice, g_2 lists b_2 as her top choice, and so on, so that b_i appears at the front of g_i's preference list for all i. If we run the Traditional Marriage Algorithm, then the resulting matching is guaranteed to match g_i to b_i for all i.

Problem 2. [A peculiar sequence] (6 points)

Define $u_0 = u_1 = 1$, and define $u_n = (n-1)u_{n-2}$ for $n = 2, 3, 4, \ldots$. By convention, $0! = 1$.

Prove that $u_{n+1}u_n = n!$ for all $n \in \mathbf{N}$.

Problem 3. [Recognizing complete binary trees] (6 points)

A *binary tree* is a tree with the property that every internal node (i.e., every non-leaf node) has exactly two children. A binary tree is called *complete* if all its leaves are at the same depth.

Assume that if `T` is a tree, then the helper function `(leaf? T)` returns true iff `T` has no children. Also, if `T` is a tree with children, then `(left T)` returns the left subtree of `T` and `(right T)` returns the right subtree of `T`. Consider the following algorithm:

```
(define (complete? T)
      (if (leaf? T)
        #t
        (and (complete? (left T)) (complete? (right T))) ) )
```

Here is an attempt to prove this algorithm correct.

Theorem 0.1: *For all trees* `T`*, if* `(complete? T)` *returns true, then there exists* $k \in \mathbf{N}$ *such that every leaf is at distance* k *from the root of* `T`*.*

Proof: Use proof by structural induction. Base case: Suppose the tree `T` has no children, so that `(leaf? T)` returns true. Then `(complete? T)` also returns true, and moreover every leaf is at distance 0 from the root (i.e., $k = 0$), so the implication is true in this case.

Inductive step: Assume the claim is true for trees `TL` and `TR`. Suppose the tree `T` is constructed so that the left subtree of the root is `TL`, and the right subtree is `TR`. Assume `(complete? T)` returns true. By the definition of `(complete? T)`, this means that `(complete? TL)` and `(complete? TR)` must both have returned true. Then, by the induction hypothesis, there is some k so that every leaf of `TL` is at distance k from the root of `TL`. Similarly, there is some k so that every leaf of `TR` is at distance k from the root of `TR`. In both cases, the leaf is at distance $k+1$ from the root of `T` (by the way that `T` was constructed from `TL` and `TR`). Also, every leaf of `T` falls into one of these two cases. Consequently, we have proven that if `(complete? T)` returns true, then there exists $k' \in \mathbf{N}$ so that every leaf of `T` is at distance k' from the root of `T` (in fact, $k' = k+1$). $\square$

This problem is continued on the following page.

Please tell us whether this proof is valid. In particular, answer each of the following questions with “Yes” or “No”, and explain your answer:
(a) Is the use of structural induction appropriate?

(b) Is the proof of the base case OK?

(c) Is the proof of the inductive step OK?

(d) Bottom line: Is the proof valid?

Problem 4. [Working with expressions some more] (8 points)

This question involves expressions generated by the following rules. (Note that rule 2. is slightly different from what you saw on a previous quiz.)

1. A single digit is an expression.
2. If E_1 and E_2 are expressions, then $E_1 E_2$ is an expression.
3. If E is an expression, then (E) is an expression.

Prove that no expression generated by the three rules above contains an open parenthesis immediately followed by a close parenthesis, i.e. "()".

Problem 5. [A variant on merge sort] (8 points)

Let F_k denote the kth Fibonacci number. Reminder: the Fibonacci numbers are defined by $F_0 = F_1 = 1$, and $F_k = F_{k-1} + F_{k-2}$ for $k \geq 2$. Consider the following variation on merge sort, which assumes that the number of elements in its list argument L is a Fibonacci number F_k.

algorithm FibMergesort(L)
{*L is a list of items from a totally ordered set, whose length is a Fibonacci number F_k*}
if L contains only 1 element, then return L
else
  divide L into L_1 (the first F_{k-1} items) and L_2 (the remaining F_{k-2} items)
  $sortedL_1 :=$ FibMergesort(L_1)
  $sortedL_2 :=$ FibMergesort(L_2)
  $sortedL :=$ Merge($sortedL_1, sortedL_2$)
  return *sortedL*

Assuming that the "divide" step in FibMergesort takes constant time (no comparisons) and Merge behaves as described in the lecture notes, identify which of the following expressions most closely matches the total number of comparisons performed by FibMergesort when initially given a list of F_k elements.

(a) $\mathbf{O}(k \log k)$

(b) $\mathbf{O}(k^2)$

(c) $\mathbf{O}(k\, F_k)$

(d) $\mathbf{O}(F_k \log k)$

(e) $\mathbf{O}(F_k^2)$

Justify your answer.