

CS 345 - Programming Languages
Spring 2008

MIDTERM #2

April 17, 2008

DO NOT OPEN UNTIL INSTRUCTED

YOUR NAME: _____

Collaboration policy

No collaboration is permitted on this midterm. Any cheating (*e.g.*, submitting another person's work as your own, or permitting your work to be copied) will automatically result in a failing grade. The Computer Sciences department code of conduct can be found at <http://www.cs.utexas.edu/users/ear/CodeOfConduct.html>

Midterm #2 (75 points)

Problem 1 (12 points)

Define the following terms:

Soundness of static analysis:

Deadlock:

No-Side-Effects (Declarative, Pure Functional) Language Test:

Eager evaluation:

Problem 2 (6 points)

Describe how reference counting could be used for garbage collection in evaluating the following Scheme expression:

```
(car (cdr (cons (cons a b) (cons c d))))
```

where *a, b, c, d* are previously defined names for cells whose reference counts are greater than 0 (*i.e.*, they do not become garbage). Assume that the final result of evaluation is not garbage, either. How many of the three `cons` cells can be garbage-collected?

Problem 3

Problem 3a (4 points)

Explain the ways in which deadlock and starvation can occur in the Dining Philosophers problem.

Problem 3b (5 points)

With N philosophers, how many distinct mutual exclusion locks are required in the Dining Philosophers problem? What guarantees does mutual exclusion provide?

Problem 4

Problem 4a (6 points)

Evaluate the following Scheme expressions:

```
(car (car (cdr (cdr (a b (c d) e (f g))))))
```

```
((lambda (f x y) (f x y)) * 2 (+ 3 2))
```

Problem 4b (6 points)

Redefine the following `let` and `let*` expressions using `lambda`, and evaluate the resulting `lambda` expressions.

```
(define x 6)
(define y 2)
(let ((x 4) (y x) (z (+ x y))) (+ x y z))
```

```
(let* ((x 4) (y x) (z (+ x y))) (+ x y z))
```

Problem 4c (6 points)

Rewrite the following function so that it is tail-recursive, or explain why it cannot be done.

```
(define length
  (lambda (lst) (if (empty? lst) 0 (+ 1 (length (cdr lst)))))
```

Problem 4d (6 points)

Rewrite the following function using `foldl/foldr`. **Be sure that the order of the result list is the same as for the original function.** You can assume a function `reverse` is already defined if you need it.

```
(define (map f lst)
  (if (empty? lst) '()
      (cons (f (car lst)) (map f (cdr lst))))))
```

Problem 5 (6 points)

What argument does John Hughes use to demonstrate that lazy evaluation provides a powerful modularization tool?

Problem 6 (6 points)

Write a Prolog program `rd` that removes all duplicates from a list. For example, the query `rd([a,c,a,c,b,b,a], X)` should return `X = [a,b,c]`.

Problem 7 (6 points)

Why is garbage collection important to logic programming languages?

Problem 8 (6 points)

What is the fundamental difference between overloading and dynamic lookup?