

CSE 413 Winter 2002 Midterm Exam

Name _____ ID # _____ Score _____

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

There are **8** questions worth a total of **95** points. Please budget your time so you get to all of the questions. *Keep your answers brief and to the point.*

You may refer to the following reference materials:

Scheme: Scheme report + course handouts (but not assignments or solutions)
Java: course Java slides + Java reference books

If you have extensive notes in your copies of the slides or handouts, please ask for a fresh, unmarked copy to use for the test.

Please wait to turn the page until you are told to begin.

CSE 413 Winter 2002 Midterm Exam

Question 1. (16 points, 2 points each part) Suppose we enter the following top-level definitions into a Scheme interpreter.

```
(define one `(1 2 3))  
(define taste `(sweet sour salt bitter))  
(define nums (list (+ 3 4) (- 6 4)))  
(define vals `(list (+ 3 4) (- 6 4)))
```

What is the value of each of the following expressions, given that the above definitions are in effect? If evaluating the expression produces an error, explain what is wrong.

- a) `(cddr taste)`
- b) `(cadr nums)`
- c) `(cadr vals)`
- d) `(car (cdr (reverse taste)))`
- e) `(list one one)`
- f) `(cons one one)`
- g) `(append one one)`
- h) `(or (< (cadr one) 10) (> (caar one 17)))`

CSE 413 Winter 2002 Midterm Exam

Question 2. (10 points) Suppose we evaluate the following expression at the top level of a scheme interpreter

```
(define x (cons '(2) (list 3 4)))
```

(a) (3 points) Draw a boxes-and-arrows diagram of this data structure.

(b) (2 points) How would a Scheme interpreter print (display) the value of `x`?

(c) (3 points) Now, suppose we evaluated the following:

```
(set-cdr! (cdr x) (car x))
```

Draw a boxes-and-arrows diagram showing this modified data structure (i.e., draw the modified data structure that `x` now refers to).

(d) (2 points) How would a Scheme interpreter print (display) this new value of `x`?

CSE 413 Winter 2002 Midterm Exam

Question 3. (9 points, 3 each)

(a) Some Scheme functions like `if` and `define` are referred to as special forms. What is the key difference between these special forms and ordinary functions like `cons`, `car`, and user-defined functions?

(b) Scheme provides several different ways to bind values to names; two of them are `define` and `set!`. What is the key difference between these two?

(c) What is the value of the following Scheme expression?

```
(map (lambda (f) (cons (f 3 2) '(4 5))) (list + * -))
```

CSE 413 Winter 2002 Midterm Exam

Question 4. (12 points) A car approaching an intersection can go one of three ways: left, straight, or right. For this problem, write a recursive scheme function `follow` that has two arguments: a description of an intersection (possibly nested), and a set of directions. The idea is that `(follow intersection directions)` should evaluate to the destination reached if the directions are followed starting at the given intersection. If the list of directions is too long and continues past the available intersections, function `follow` should return `#f`.

Directions are given by a list consisting of the symbols `l`, `s`, and `r`, meaning turn left, go straight, or turn right. The list `(r)` means turn right; the list `(s r l)` means go straight at the current intersection, turn right at the next one, and turn left at the last one.

An intersection is represented by a list (left straight right). Each choice is either a destination if it is a simple symbol, or the description of the next intersection in the given direction. Examples:

`(follow '(a b c) '(r)) => c` (i.e., turning right reaches `c`)

`(follow '(a (b c (d e f)) g) '(s r l)) => d`
(i.e., straight to `(b c (d e f))`, right to `(d e f)`, left to `d`)

`(follow '(a (b c (d e f)) g) '(s r)) => (d e f)`
(i.e., straight to `(b c (d e f))`, right to `(d e f)`)

`(follow '(a b (x y z)) '(r l r l)) => #f`
(too many directions: right to `(x y z)`, left to `x`, but can't go further)

CSE 413 Winter 2002 Midterm Exam

Question 5. (12 points) The standard Scheme map function produces its result by applying a given function to each element of a list

```
(map (lambda (n) (+ n n)) '(2 -6 4 0 9)) => (4 -12 8 0 18)
```

Function map cannot, however, be used to apply a function to values in a nested list.

For this problem, write a function `deep-map` that applies a function to each element of a possibly nested list data structure, and returns a result that has the same shape as the original list argument.

```
(deep-map (lambda (n) (+ n n)) '((2 -6) 4 (((0)) 9))) =>
((4 -12) 8 (((0)) 18))
```

CSE 413 Winter 2002 Midterm Exam

Question 6. (12 points) If you recall from lecture, the Fibonacci numbers are defined using as the sequence

$$\begin{aligned}\text{fib}(0) &= 0 \\ \text{fib}(1) &= 1 \\ \text{fib}(n) &= \text{fib}(n-1) + \text{fib}(n-2)\end{aligned}$$

In other words, each number in the sequence is the sum of the previous two: 0, 1, 1, 2, 3, 5, 8, 13, When we coded this up directly as a Scheme function, we discovered it ran very slowly for even moderate values of n , because its execution time was an exponential function of n .

For this problem, write a properly *tail-recursive* function (`fib-t n`) that computes the n^{th} Fibonacci number. For full credit, your function should run in linear time as a function of n (i.e., the number of recursive calls should be proportional to n .) You will probably need an auxiliary (helper) function in addition to function `fib-t`.

CSE 413 Winter 2002 Midterm Exam

Question 7 (cont).

- c) In Java, a new type can be defined in two ways: by defining a class or by defining an interface. What are two fundamental differences between classes and interfaces?

- d) One of your colleagues, A. Hacker, is working on the bank's automatic teller machine software. Individual bank accounts are represented by instances of a class `BankAccount`. Hacker is an old-time Fortran programmer who is new to Java and isn't comfortable yet with containers like `ArrayList`. So his/her code uses arrays instead. The following code is supposed to create an array of `BankAccounts` and set their balances to 0.

```
BankAccount[ ] accounts = new BankAccount [10];
for (int k = 0; k < accounts.length; k++) {
    accounts[k].setBalance(0.0);
}
```

There is a `setBalance` method in class `BankAccount`, which can be used to change the account balance. But when this code is executed, a `nullPointerException` is thrown. What went wrong?

CSE 413 Winter 2002 Midterm Exam

Question 8. (12 points, 2 each) Every Java application begins execution in a main method that has the following signature (i.e., function heading)

```
public static void main (String[ ] args) { ... }
```

The question is, what does all that gibberish mean anyway? Below, give a brief definition or explanation of the meaning of each keyword or other item in the heading.

public

static

void

main

String[]

args