University of California, Berkeley – College of Engineering

Department of Electrical Engineering and Computer Sciences

Spring 2001 Instructor: Dan Garcia 2001-02-26

CS 3 Midterm #1

Personal Information

Last name	
First Name	
Student ID Number	
*Lab Section Time & Location **you attend***	
All the work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CS3 who have not taken it yet. (please sign)	

Instructions

- Question 0 (worth 1 point) Please fill in the front and write your name on every page!
- You have two hours to complete this quiz. The quiz is open book and open notes, no computers.
- Partial credit will be given for incomplete / wrong answers, so please write down as much of the solution as you can***.***
- You may always write auxiliary functions for a problem unless they are specifically prohibited in the question.
- Feel free to use any Scheme function that was described in sections of the textbook we have read without defining it yourself. Do not use functions or constructs that we have not covered, such as recursion.
- You do not need to write comments for functions you write unless you think the grader will not understand what you are trying to do otherwise.
- Please comment on the exam on the right. Rate its difficulty (0 = cake, 5 = impossible), fairness (0 = unfair, 5 = fair), and feel free to add any other comments that come to mind.

Name: _____

Grading Results

Question	*Max. Points*	*Points Given*
0	**1**	
1	**20**	
2	**10**	
3	**8**	
4	**8**	
5	**13**	
Total	**60**	

Comments:

- Difficulty (0=easy, 5=hard):
- Fairness (0=unfair, 5=fair):
- Other thoughts?

Name: _____

Question 1 : You're in Big Momma's house now… (20 points)

Your grandmother tells you that back in her day they didn't have those fancy computers to evaluate all of their computer programs for them. No, they evaluated all of their programs themselves. Prove to your grandmother that you could do the same if you had to. For each of the following Scheme expressions, write the value that would be returned if we were to evaluate it in the Scheme interpreter. Write your answers next to each expression in the space provided. If an expression results in an error, just write the word ERROR and describe the error in a few words. If the value of an expression is a procedure, just say PROCEDURE. Assume that the following `defines` have already been made:

```
(define mantra '(cal is great) )

(define (weird a b c) '(/ 100 a) )

(define (mult-10 n) (word n 0) )
```

CAUTION: These are quite subtle; make sure you go through them carefully.

`(* (+ 2 (/ 7 1)` `(*) ) )`	
`(equal? (quote (1)) (se (1)) )`	
`(if (= 1 '1) + -)`	
`(if < 3 4)`	

`(and (if #f #f #t)`
`'a`
`'(b)`
`(not (= 1 2))`
`mantra)`	

`(bl (bf mantra))`	
`(lambda () m a n 't 'r 'a)`	
`(weird 0 1 2)`	
`(every mult-10 '(weird 10 2 3) ))`	
`(accumulate - (every mult10 '(3 2 1)) )`	

Name: _____

Question 2 : My broker told me of this great (mutual) fun… (10 points)

Given the following definition:

```
(define mutual-fun
  (lambda (f g)
    (if (< (f 1)
           (g 1))
        f
        g)))
```

a) Describe, **as precisely as possible**, the domain and range of `mutual-fun`. (6 points)

Domain	Range

b) Below is an expression with a single call to `mutual-fun` with some parenthesis and arguments missing. Fill in the missing parenthesis and arguments so that the expression returns `2`. (4 points)

```
mutual-fun sqrt 4
```

Name: _____

Question 3 : $35 for dinner, and nothing… (8 points)

We've begun to write an error checking function, called `bad-date?` for the case-study. It takes in a date, which is a sentence, and makes sure it is correctly formatted. However, the function is incomplete, as it misses an important error. We need you to fix it with an error check <u>in the correct place</u> in the `cond`.

- Fill in <u>only one</u> of the appropriate boxes below corresponding to the missing error-check. Be sure to give a suitable return value (i.e., a self-explanatory word) for the missing check.
- Use the appropriate selector functions as in the case study.

Here are examples of calls to `bad-date?` :

```
: (bad-date? '(january 5 1981)) ➔ wrong-number-of-elements-in-date
: (bad-date? '(march fourth))   ➔ day-is-not-an-integer
: (bad-date? '(dec 25))         ➔ illegal-month-supplied
: (bad-date? '(february 26))    ➔ #f

(define legal-month-names '(january february march april may june july august
                            september october november december))))

(define (bad-date? date)
          (cond ((not (= (count date) 2))
           'wrong-number-of-elements-in-date)
```


```
          ((not (integer? (day-in-month date)))
           'day-is-not-an-integer)
```


```
          ((not (member? (month-name date) legal-month-names))
           'illegal-month-supplied)
```


```
          (else #f)))
```

Name: _____

Put down your pen or pencil, stretch, take a deep breath, and proceed…

Name: _____

Question 4 : Waiter, there's a bug in my function! (10 points)

Assume you want to sum up the squares of every number in a sentence. You write the following procedure (formatted to make it easy to pinpoint bugs):

```
  (define (sum-of-squares sent)
   (accumulate
1              (lambda
2                      (total newone)
3                      (+
4                         total
5                         (* newone newone)))
6               sent))
```

a) Typing `(sum-of-squares '(1 2 3))` gives you `122` when it should give you `1*1 + 2*2 + 3*3 = 14`. You look at the code for a minute and say, "Ah!" and make a small change. The next time, `(sum-of-squares '(1 2 3))` gives you `8`. Fill in the blanks below to specify the change you must have made. (4 points)

Replacing line # ____,

with _____

would cause `(sum-of-squares '(1 2 3))` to return `8` instead of `122`.

b) Since `8` still isn't the answer you want, you make **another** change. This time, `sum-of-squares` works, and returns `14` when given the sentence `(1 2 3)`. What did you change? Fill in the blanks below to specify the change you must have made. (4 points)

Replacing line # ____,

with _____

would cause `(sum-of-squares '(1 2 3))` to correctly return `14` instead of `8`.

Name: _____

Question 5 : Magical mystery function, step right this way… (13 points)

Let's say that we have the following two definitions:

```
(define (add-one x)
  (+ x 1))

(define (mystery L)
  (se 1 (every add-one L)))
```

a) What does `(mystery '())` evaluate to? (1 point)

b) What does `(mystery (mystery '()))` evaluate to? (2 points)

c) Write a function, `1-to-b`, that takes a positive integer `b`, and returns a sentence of numbers from `1` to `b`, inclusive. (5 points) Here are sample calls:

```
: (1-to-b  1) ➔ (1)
: (1-to-b 10) ➔ (1 2 3 4 5 6 7 8 9 10)
```

d) Write a function, `a-to-b`, that takes positive integers `a` and `b`, returns a sentence of numbers from `a` to `b`, inclusive. You may assume `(<= 1 a b)`. You should call `1-to-b` in your solution, and you shouldn't use any function you called in the body of `1-to-b` above. (5 points) Here are sample calls:

```
: (a-to-b 7  7) ➔ (7)
: (a-to-b 7 10) ➔ (7 8 9 10)
```