CS3 Spring 05 – Midterm 1

Read and fill in this page now

Your name:	________________
Your login name:	________________
Your lab section day and time:	________________
Your lab T.A.:	________________
Name of the person sitting to your left:	________________
Name of the person sitting to your right:	________________

Prob 0:	________	Prob 1:	________	________
Prob 2:	________	Prob 3:	________	
Prob 4:	________	Prob 5:	________	________

Subtotal: _________/42 Scaled Total: __________/28

You have 50 minutes to finish this test, which should be reasonable; there will be approximately 20 minutes of leeway given past one hour. Your exam should contain six problems (numbered 0-5) on 9 total pages, along with the code from the "Difference Between Dates version 2" case study at the end. Note that the problems at the end of this exam may take longer than those at the front!

This is an open-book test. You may consult any books, notes, or other paper-based inanimate objects available to you. Read the problems carefully. If you find it hard to understand a problem, ask us to explain it. If you have a question during the test, please come to the front or the side of the room to ask it.

Restrict yourself to Scheme constructs covered in chapters 3-6 , 11-13 of *Simply Scheme*, the "Difference Between Dates" case study, parts 1 and 2, and the "Roman Numerals" case study.

Please write your answers in the spaces provided in the test; if you need to use the back of a page make sure to clearly tell us so on the front of the page. We believe we have provided more than enough space for your answers, so please don't try to fill it all up.

Partial credit will be awarded where we can, so do try to answer each question.

Relax!

Problem 0. (1 point, 1 minute)

Put your login name on each page. Also make sure you have provided the information requested on the first page.

Problem 1. ***2-parts*** **(6 / 4 points, 10 minutes) : Make test cases**

A procedure called `starts-or-ends-with?` (which you can abbreviate as `soew?`) was written by a classmate:

Specification `(soew? letter wd)`
Input: `letter` is a word of length 1; `wd` is a word of any length
Output: `#t` if the letter is at the beginning or at the end of the word, `#f` otherwise

(a) Provide enough test cases that will assure that it works correctly, given the specification above. Do not provide too many test cases, though! Note that you don't know how your classmate wrote `soew?`, so minimize the assumptions you make about how the code is written (although, you will have to assume some things).

You don't need to test for situations outside of the specification (e.g., where the second parameter is a sentence, or not given in a call). You are testing that the *logic* of the program is right.

(a) Now write `starts-or-ends-with`.

Problem 2. (6 points, 8 minutes): Make the expressions correct

Fill in the blanks such that the expression evaluates to the specified value (i.e., what is shown after the arrow). You can leave some blanks unfilled, but cannot ignore text that is present. If no such entry will create the proper evaluation, write the word IMPOSSIBLE to the right of the question.

1]	`(day-span '(january 3) ____________________ )` ➔ 1
2]	*[[for this problem, do not use quotes in text you enter]]* `(____________________ '(can you hear me) ________________)` ➔ ou

3]	`(and (equal? 1 ____________)`
	`     (or (equal? 1 2) (equal? 1 3) (equal? 1 4))`
	`     (equal? 1 1))`
	➔ #t

4]	`(define (vowel? ltr)                  ;; these functions`
	`   (member? ltr '(a e i o u)))        ;; are define for the`
	`                                      ;; next three`
	`(define (mystery wd)                  ;; questions`
	`  (if (empty? wd)`
	`      #t`
	`      (if (vowel? (first wd))`
	`          #f`
	`          (mystery (bf wd)))))`
	
	`(mystery 'strength)` ➔ ________________

5]	`(mystery "")` ➔ _________________
6]	`(mystery '(this is scheme fosho))`➔ _____________________

Problem 3. (6 points, 8 minutes): Count adjecent duplicates

Write a procedure `count-adjacent-duplicates` (`cad`) that takes as input a sentence of any length, and returns a numeric count of the how many adjacent members of the sentence are the same.

`(cad '(I had had enough!))`	→	1
`(cad '(a a a a))`	→	3
`(cad '(1))`	→	0

Problem 4. (7 points, 10 minutes): Backwards day-span…

Write a function `date-in-year` which, in some ways, does the reverse of `day-span`. `date-in-year` takes a numeric span which is the number of days after January 1st in a non-leap year, and returns the date that is defined by that number of days.

`(date-in-year 5)`	➔	`(january 5)`
`(date-in-year 34)`	➔	`(february 3)`
`(date-in-year 325)`	➔	`(november 21)`

You may find the code to the Difference between Dates version 2 case-study useful; it is reproduced in an appendix at the end of this exam. Note that you will receive partial credit for solutions that work for some of the months.

Problem 5. *2-parts* (8 / 4 points, 13 minutes):
Transform song titles: help a friend with your new powers

A friend knows that you have extensive programming skills, having taken 6 weeks of CS3, and asks if you can help with a problem. He had a big list of song titles on his computer and they somehow got corrupted; he wants you to fix the titles. The titles are words in scheme, without spaces, which is the way your friend wants it. However, there are problems with some of the titles:

- It seems that, sometimes, a title starts with `%20`: that should be removed if present.
- Also, if the title doesn't end in a period, something was cut off: you should add three periods "`...`" at the end to indicate that.
- Finally, some titles have garbage letters in them – either `$`, `&`, or `@` -- if the title has that then the whole thing should be thrown away (turned into an empty word).

The shortest title is 10 letters long, but most are much longer.

`(fix-title '%20JimmyHasAF)`	➔	`JimmyHasAF...`
`(fix-title 'WhenI$WasAChild.)`	➔	`""`
`(fix-title '%20Tom@and^^zaaz)`	➔	`""`
`(fix-title 'SouthMainStreet.)`	➔	`SouthMainStreet.`

(a) Write a procedure `fixed-title` that takes the title (as a word) and returns a fixed version of it, as specified above. You should define the helper procedures that are described below, using good names for the procedures and parameters, and use the helper procedures in `fixed-title`.

```
;;returns true if the title should be thrown away
(define (garbage-title? title)
```

```
;; returns true if the front of the title has a problem
(define
;; returns true if the end of the title was cut off
(define
```

```
;; returns the fourth element and on of the word
(define (but-first-three x)
   (bf (bf (bf x))))
```

```
;; fixed-title
(define (fixed-title title)
```

(b) Your friend has lots and lots of titles, though; you will need to automate things. Write a procedure `all-titles-fixed` which takes a sentence of titles (each as a word), of any length, and returns a sentence where each title is fixed.

(You can still get full credit on this problem if you haven't answered part (a) correctly)

Appendix A: Difference Between Dates, part I, version 2 code.

This is not a question!

```
; Access procedures for the components of a date.
(define (month-name date) (first date))
(define (date-in-month date) (first (butfirst date)))

; Return the number of days from January 1 to the first day
; of the month named month.
(define (days-preceding month)
  (cond
    ((equal? month 'january) 0)
    ((equal? month 'february) 31)
    ((equal? month 'march) 59)
    ((equal? month 'april) 90)
    ((equal? month 'may) 120)
    ((equal? month 'june) 151)
    ((equal? month 'july) 181)
    ((equal? month 'august) 212)
    ((equal? month 'september) 243)
    ((equal? month 'october) 273)
    ((equal? month 'november) 304)
    ((equal? month 'december) 334) ) )

; Return the number of days from January 1 to the given
; date, inclusive.  Date represents a date in 2002.
(define (day-of-year date)
  (+
    (days-preceding (month-name date))
    (date-in-month date)) )

; Return the difference in days between earlier-date and
; later-date. Earlier-date and later-date both represent
; dates in 2002, with earlier-date being the earlier of
; the two.
(define (day-span earlier-date later-date)
  (+
    1
    (-
      (day-of-year later-date)
      (day-of-year earlier-date) ) ) )
```