<!--course, exam #, semester/year (e.g., CS 150, Midterm #2, Fall 1994)-->

CS3, Spring 97
Midterm1
Professor Mike Clancy

(50 minute open-book exam, 30 total points, 4 problems)

Problem #1 (4pts, 8min)

Write a function named rating that, given an integer n as its single argument, returns one of the atoms BAD, GOOD, or OK as follows:

situation	*return value*
n > 110	GOOD
n > 60 and n < 111	OK
n < 61	BAD

Your function should make as few comparisons as possible.

Problem #2 (6pts, 8min)

Clearly fill in parentheses and quotes so that the expressions below return the list (march 3 (april 27))

<part a>

```
cons    march    3    april    27
```

<part b>

```
append    march    3    april    27
```

Problem #3 (8pts, 18min)

<part a>

Write a function named wednesday-span that, given as arguments two dates in 1997 in the format described in the "Difference Between Dates" case study, returns the number of Wednesdays in the period spanned by the two dates. January 1, 1997 was a Wednesday (as December 31 will be). Some examples appear below.

expression	*desired result*
(wednesday-span '(january 2) '(january 5))	0
(wednesday-span '(january 6) '(january 9))	1
(wednesday-span '(january 1) '(january 8))	2
(wednesday-span '(january 1) '(december 31))	53

You may assume that the first argument date is the same as or earlier than the second argument date. You may also define and use auxiliary functions, and use any function appearing in the "Difference Between Dates" code (which works as well for dates in the 1997 as for dates in 1994) without defining it.

<!--course, exam #, semester/year (e.g., CS 150, Midterm #2, Fall 1994)-->

<part b>
Does your solution more closely resemble the first approach taken in the "Difference Between Dates" case study--i.e. the approach used to design the program in Appendix A--or the second approach? Briefly explain (i.e. in one or two sentences).

Problem #4 (10pts, 15min)

Your lab partner accidentally changes on of the symbols (words or numbers) in the code from Appendix A of the "Difference Between Dates" case study. You find, when using the accidentally modified code, that a call to day-span with arguments (october 1) and (november 1) returns 1 instead of the correct answer 32.

<part a>
With as general a statement as possible, complete the following sentence:

"Your lab partner's change has made the program erroneously act as if ...

<part b>
The code from Appendix A appears on the last two pages of this exam. Specify a possible cause of the bug, that is, a substitution of a single symbol that would produce the behavior described.

function *substitution* *position in the function*

<part c>
Specify a second possible cause of the bug that's in a different function from the bug you specified in <part b>.

function *substitution* *position in the function*

<part d>
Provide an expression which, when evaluated, would allow you to rule out one of the causes you described in parts b and c. Also indicate how you would use the result of evaluating the expression to determine which of the two possible bug causes to eliminate.

Appendix A--Partially designed attempt to compute the difference between dates

```
; return the number of days spanned by earlier-date and later-date.
; earlier-date and later-date both represent dates in 1994,
; with earlier-date being the earlier of the two.
; note:  general-day-span is not implemented.
; ed = earlier-date, ld = later-date
```

<!--course, exam #, semester/year (e.g., CS 150, Midterm #2, Fall 1994)-->

```
(define (day-span ed ld)
    (cond
        ((same-month? ed ld)
            (same-month-span ed ld) )
        ((consecutive-months? ed ld)
            (consec-months-span ed ld) )
        (else (general-day-span ed ld) ) ) )


; access functions for the components of a date.

(define (month-name date) (car date))
(define (date-in-month date) (cadr date))


; return true if d1 and d2 are dates in the same month, and false otherwise.  d1 and d2 are dates in 1994.

(define (same-month? d1 d2)
    (equal? (month-name d1) (month-name d2)))


; return the number of the month with the given name.

(define (month-number m)
    (cond
        ((equal? m 'january) 1)
        ((equal? m 'february) 2)
        ((equal? m 'march) 3)
        ((equal? m 'april) 4)
        ((equal? m 'may) 5)
        ((equal? m 'june) 6)
        ((equal? m 'july) 7)
        ((equal? m 'august) 8)
        ((equal? m 'september) 9)
        ((equal? m 'october) 10)
        ((equal? m 'november) 11)
        ((equal? m 'december) 12) ) )


; return true if d1 is in the month that immediately precedes the month d2 is in, and false otherwise.
; d1 and d2 are dates in 1994.

(define (consecutive-months? d1 d2)
    (=
        (month-number (month-name d2))
        (+ 1 (month-number (month-name d1))) ) )


; return the difference in days between ed and ld, which are dates in the same month in 1994.
```

<!--course, exam #, semester/year (e.g., CS 150, Midterm #2, Fall 1994)-->

```
(define (same-month-span ed ld)
    (+ 1 (- (date-in-month ld) (date-in-month ed)) ) )


; return the number of days in the month named m

(define (days-in-month m)
    (cond
        ((equal? m 'january) 31)
        ((equal? m 'february) 28)
        ((equal? m 'march) 31)
        ((equal? m 'april) 30)
        ((equal? m 'may) 31)
        ((equal? m 'june) 30)
        ((equal? m 'july) 31)
        ((equal? m 'august) 31)
        ((equal? m 'september) 30)
        ((equal? m 'october) 31)
        ((equal? m 'november) 30)
        ((equal? m 'december) 31) ) )


; return the number of days remaining in the month of the given date including the current day.  date is in
1994.

(define (days-remaining date)
    (+ 1 (- (days-in-month (month-name date)) (date-in-month date))) )


; return the difference in days between ed and ld, which are dates in consecutive months in 1994.

(define (consec-months-span ed ld)
    (+ (days-remaining ed) (days-in-month ld)))
```
