Informatics 41 • Fall 2005 • David G. Kay • UC Irvine Your name _____

THIRD QUIZ

Your student ID _____

You have 15 minutes from the start of class to complete this quiz. The usual instructions apply. Don't reinvent the wheel–for full credit, use functions from an earlier problem where helpful in a later one. Good luck!

You may show lists in any of three ways: `(list 73 15)`, `'(73 15)`, or `(cons 73 (cons 15 empty))`.

Problem 1 (4 points)

Evaluate each of the following expressions. Use this definition independently for each of the four parts:

```
(define L (cons 'leaves (cons 'sweaters (cons 'rain (cons 'turkey empty)))))
```

(a) `(first L)`

(b) `(rest L)`

(c) `(first (rest L))`

(d) `(rest (rest L))`

(e)
```
(cond
      ((empty? (rest (rest (rest L)))) 'Maine)
      ((empty? (rest (rest L))) 'Vermont)
      (else 'NewHampshire)))
```

Problem 2 (2 points)

What three lines does DrScheme produce when it evaluates this code?

```
(define election-years (cons 2000 (cons 2004 (cons 2008 empty))))
(rest election-years)
(first election-years)
(rest election-years)
```

Problem 3 (4 points)

Complete the definition of `count-items-on-list` below.

```
;; count-items-on-list:  expression list -> number
;; Return the number of times the expression occurs on the list
(define count-items-on-list
  (lambda (item L)
      (cond
          (_____________________ 0)
          ((equal? item _____________________) (+ 1 (count-items-on-list item (rest L))))
          (else (_____________________  item  _____________________)))))
```

Problem 4 (14 points)

A time is a structure `(make-time ampm hour minute)` where `ampm` is a symbol (`'am` or `'pm`), `hour` is a number from 0 to 11, and `minute` is a number from 0 to 59:

```
(define-struct time (ampm hour minute))
```

(a) (3 points) The time in Pakistan is 12 hours different from the time in Irvine. That means we can convert Pakistan time to Irvine time (and vice versa) simply by changing `'am` to `'pm` or `'pm` to `'am` in the first field of the structure. Define the function `switch-time` that takes a time and returns a time, changed in that way.

```
;; switch-time: time -> time
;; Return time with first field changed (am->pm, pm->am)
(define switch-time
  (lambda (T)
```

(b) (5 points) Define the function `switch-time-list` that changes a list of Irvine times to Pakistan times.

```
;; switch-time-list:  list-of-times -> list-of-times
;; For each time in the list, change am to pm and vice versa.
(define switch-time-list
     (lambda (L)
```

(c) (6 points) Suppose we already have defined the function `time<?` ("time less-than") that takes two times and returns true if the first time is earlier than the second time (and false otherwise).

Define the function `remove-earlier-times` that removes from a list of times all the items that are earlier than a specified cutoff time. Of course you should call `time<?` where necessary.

```
;; remove-earlier-times:  list-of-times  time  -> list-of-times
;; Return a list containing all (and only) the times from the first input (a list of
;;   times) that are NOT earlier than the second input (the "cutoff time")
(define remove-earlier-times
     (lambda (L cutoff)
```