Informatics 41 • Fall 2004 • 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. Read the questions with care; work with deliberate speed. Don't give us more than we ask for. The usual instructions apply. Good luck!

Problem 1 (4 points)

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

(a) `(first (cons 'Ohio (cons 'Iowa (cons 'Idaho empty))))`

(b) `(rest (cons 'Ohio (cons 'Iowa (cons 'Idaho empty))))`

(c)
```
(define L1 (cons  Lanai (cons  Molokai (cons  Oahu (cons  Hawaii empty)))))
   (cond
      ((equal? (first L1) 'Hawaii) 'Niihau)
      ((equal? 'Oahu (rest (rest L1))) 'Kauai)
      (else (first (rest L1)))))
```

Problem 2 (4 points)

Complete the definition of `item-on-list?` below.

```
;; item-on-list?:  expression list -> boolean
;; Return true if the expression occurs on the list
(define item-on-list?
    (lambda (item L)
        (cond

            (_____________________ false)

            ((equal? item _____________________) true)

            (else (_____________________  item  _____________________)))))
```

Problem 3 (17 points)

A date is a structure `(make-date month day year)` where `month` is a symbol (`'Jan`, `'Feb`, and so on), `day` is a number from 1 to 31, and `year` is a number from 1000 to 3000:

```
(define-struct date (month day year))
```

(a) (2 points) Suppose we have this function to check whether a date is valid:

```
;; valid-date?: anything -> boolean
(define MONTHLIST (list 'Jan 'Feb 'Mar 'Apr 'May 'Jun 'Jul 'Aug 'Sep 'Oct 'Nov 'Dec))
(define valid-date?
  (lambda (X)
    (and (date? X)  ; Is it a date structure in the first place?
         (item-on-list? (date-month X) MONTHLIST)
         (>= (date-day X) 1) (<= (date-day X) 31)
         (>= (date-year X) 1000) (<= (date-year X) 3000))))
```

Give two examples of dates in the year 2004 or 2005 that are incorrect but *not* caught by this function.

(b) (6 points) Define the function `keep-valid-dates` that removes invalid dates from a list.

```
;; keep-valid-dates:  list -> list-of-dates
;; Return a list containing all (and only) the valid dates from the input list
(define keep-valid-dates
    (lambda (L)
```

(c) (9 points) Define the function `update-year` that changes one year to another in a list of dates.

```
;; update-year:  number number list-of-dates -> list-of-dates
;; For each date in the list, if its year matches the first number, change it to the
;; second number; otherwise, leave the date unchanged.
(define update-year
    (lambda (oldyear newyear L)
```