SEVENTH QUIZ

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 (6 points)

What is the value of each of the following expressions? (Remember that odd? returns true if its input is an odd number; second returns the second item on a list ((first (rest L)); and sub1 subtracts 1 from its argument.)

(a) (filter odd? (list 41 42 43 44 45))

(b) (filter name-value? L)

(c) (map sub1 (list 17 18 19 20))

(d) (map (lambda (N) (* N 5)) (list 2 3 4 5))

(e) (foldr * 1 (list 10 4 3))

(f) (foldr (lambda (L1 L2) (append L1 L2))
  empty
  (list '(1 2) '(3 4 5) '(6 7 )))

Problem 2 (16 points)

Suppose we have a list called BL of books defined as follows:

(define-struct book (title author genre price sold instock))

where title and author are strings, genre is a symbol (e.g., 'cookbook or 'humor) representing the category of the book, price is a number representing the price of one copy, sold is the number of copies sold, and instock is the number of copies in stock.
Here’s another copy of the structure definition, so you don’t have to keep flipping the page over:

(define-struct book (title author genre price sold instock))

(a) (6 points) For each of the following expressions, describe in one clear and precise English phrase what value it returns. Don’t just say, “It does a foldr of plus and zero and ...”; give a description of what the expression means, something you could put in a software catalog so that a prospective buyer could find what he or she wanted. Use real-world terms, not program syntax terms: Say something like, “a list of the authors whose books earned over $1,000,000,” not “books whose book-sold field is greater than 1000.”

(a.1) (map book-author (filter (lambda (B) (< (book-price B) 10.00 )) BL))

(a.2) (local ((define chosen (filter (lambda (B) (equal? (book-genre B) 'travel)) BL)))
// (foldr + 0 (map (lambda (B) (* (book-price B) (book-sold B))) chosen))
(length chosen)))

(b.1) (2 points) Define the function book-matches-genre? as described below. [Hint: Use member?]

;; book-matches-genre?: book list-of-symbols -> boolean
;; Return true if the book’s genre appears on the input list of symbols.
;; EXAMPLE: (book-matches-genre? B (list ‘poetry ‘drama ‘fiction)) returns true if
;; B is a poetry book and false if B is a sports book.
(define book-matches-genre?
  (lambda (B genrelist)

(b.2) (3 points) Using map, filter, and/or foldr, define the following function without using explicit recursion. Use previously defined functions where possible.

;; selected-genre-books: list-of-books list-of-symbols -> list-of-books
;; Return a list of those books on the input booklist whose genre appears on the input
;; list of symbols.
(define selected-genre-books
  (lambda (booklist genrelist)

(b.3) (5 points) Using map, filter, and/or foldr, define the following function without using explicit recursion. Use previously defined functions where possible.

;; available-titles-by-genre: list-of-books list-of-symbols -> list-of-string
;; Return a list of the titles of books on the input booklist that (a) match one of
;; genres on the input list of symbols and (b) have at least one copy in stock.
(define available-titles-by-genre
  (lambda (booklist genrelist)