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 (3 points)
What is the value of each of the following expressions? (Remember that even? returns true if its input is an even number, second returns the second item on a list, and add1 adds 1 to its argument.)


(a) (filter even? (list 1 3 2 4 7 8))

(b) (map add1 (list 4 6 8 10 12))

(c) (map (lambda (N) (/ N 10)) (list 9000 700 50))

(d) (foldr (lambda (P L) (cons (second P) L)) empty LO)

Problem 2 (13 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 string (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.

(a) (4 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-title (filter (lambda (B) (= 0 (book-instock B))) BL))

(a.2) (first (quicksort BL (lambda (B1 B2) (> (book-price B1) (book-price B2)))))
(b) (4 points) Using map, filter, and/or foldr, define the following function without using explicit recursion.

;; books-by-author: list-of-book string -> list-of-book
;; Return a list of books in the input list written by the specified author.
(define books-by-author
  (lambda (L name)
    (filter (lambda (B) (string=? name (book-author B))) L)))

SCORING: 1 point for filter … book-author … L
1 point for correct comparison of string parm with book-author (equal? is fine)
1 point for syntactically correct lambda expression in filter (using local or helper is OK)
1 point for everything else correct.

(c) (5 points) Using map, filter, and/or foldr, define the following function without using explicit recursion.

;; genre-average-price: list-of-books string -> number
;; Return the average price of books of the specified genre
(define genre-average-price
  (lambda (booklist genre)
    (/ (foldr + 0 (map book-price (filter (lambda (B) (string=? genre (book-genre B))) booklist)))
      (length (filter (lambda (B) (string=? genre (book-genre B))))))))

SCORING: 1 pt for returning result of a division (with some attempt at totalling and counting books
1 pt for filtering booklist for the specified genre
1 pt for properly summing up total price (including map and foldr)
1 pt for properly counting relevant books (using a local to avoid the duplicate work isn’t required)
1 pt for everything else correct.

Problem 3 (4 points)
We have distributed a Deus X reference sheet; you don’t have to turn it back in, but we’ll recycle it if you do. We’ll have a better chance of assigning you partial credit if you show your work (e.g., draw a picture of the register(s) and/or memory locations).

Suppose that location 333 of the Deus X machine’s memory holds the word LION, that location 444 holds the word BEAR, and that location 555 holds the word SEAL. What does the Deus X machine print after executing these instructions?

0. 10 333 (lda 333)
1. 50 555 (cmpa 555)
2. 62 5 (jg 5)
3. 6 555 (out 555)
4. 7 6 (jmp 6)
5. 6 444 (out 444)
6. ...