SECOND QUIZ

You have 15 minutes from the start of class to complete this quiz. Read the problems 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)

Evaluate each of the following expression(s). (That is, what does DrScheme display in the interactions window when you enter the expression(s) in the definitions window and click Run?)

(a) (define MESSAGE "Go Eaters")
   (string-length (string-append MESSAGE "!!!"))

(b) (define shipping-charge
   (lambda (weight)
     (cond
       ((<= weight 12) 7.95)
       ((<= weight 24) 9.95)
       ((<= weight 36) 10.95)
       (else (+ 10.95 (* 0.50 (+ 1 (/ (- weight 36) 12))))))
   )
   (shipping-charge 30)

(c) (*
    (+ 5 4 3)
    (/ (* 25 2) (/ 20 4)))

Problem 2 (6 points)

The Anteater Coffee House uses this structure to store information about each drink a customer orders:

(define-struct order (drink size price customer))

where drink (a string) is the name of the drink; size (a string) is “small”, “medium”, or “large”; price (a number) is the price; and customer (a string) is the name of the customer who made the order.

(a) (2 points) Write an expression that constructs and returns a structure representing a medium espresso ordered by Sam that costs $2.25. (Just write the expression that constructs the structure; don’t use define.)

(b) (1 point) What is the name of the selector function that takes an order structure and returns how big the ordered drink is?

(continued on reverse)
(c) (3 points) Fill in the body of the function defined below.

```scheme
;; drink-ordered-by?: order string -> boolean
;; Return true if the drink was ordered by the named person (and false otherwise)
;; Example: (drink-ordered-by? (make-order “latte” “small” 3.50 “Joe”) “Jane”)
;; should return false.
(define drink-ordered-by?
 (lambda (this-order name-to-check)
    (string=? (order-customer this-order) name-to-check)))
```

**Problem 3** (11 points)

The national cellphone service in Outer Yucca charges a lot for text messages: $3.50 plus $0.10 for each character in the message. Write the function `text-cost` that takes a string (the message) as its input and returns a number, the cost of that message according to the pricing described above.

Write a contract, a brief purpose statement, two examples/tests in the form of `check-expect` expressions, and the Scheme function definition. Use the following constants in your code:

```scheme
(define FIXED-PRICE 3.50)
(define PRICE-PER-CHAR 0.10)
```

```scheme
(check-expect (text-cost “Hello”) 4.00)
(check-expect (text-cost “”) 3.50)  ;; [2 points total for tests; pretty much any test is OK; don’t deduct for
;; minor flaws in check-expect]
(define text-cost ;; [SCORING for code: see below]
 (lambda (message)
    (+ FIXED-PRICE (* PRICE-PER-CHAR (string-length message)))))  ;; or arithmetically equivalent expression
```