FOURTH 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 (20 points)

The Anteater Fruit Stand represents each kind of fruit it sells using a structure defined as follows:

```
(define-struct fruit (name price received sold))
```

The name field is the kind of fruit (a string), price is how many dollars per pound the stand charges for this product, received is the number of pounds delivered to the stand by the growers, and sold is the number of pounds the stand has sold.

(a) (2 points) Write an expression (not a function definition) that constructs and returns a structure showing that the stand sells oranges at $2.50 per pound, has had 125 pounds delivered, and has sold 80 pounds.

(b) (1 point) If `F` is a fruit structure, write an expression (not a function definition) that returns the price per pound of `F`.

(c) (3 points) Complete the definition of the function `fruit-revenue` as described below.

```
;; fruit-revenue:  fruit  ->  number
;; Return the total amount of money we have earned from selling this fruit
(define fruit-revenue
  (lambda (F)
```

(d) (5 points) When a grower sends us more of an item, we need to update our records. Complete the definition of the function `new-shipment` as described below.

```
;; new-shipment:  fruit  number  ->  fruit
;; Return the fruit structure updated to show the additional number of pounds received
(define new-shipment
  (lambda (F pounds)
```

(e) (4 points) Sometimes multiple shipments result in the fruit stand's database having more than one structure for the same fruit. Complete the definition of the function `same-fruit?` as described below. [Hint: Use `and`.]

```
;; same-fruit?:  fruit  fruit  ->  boolean
;; Return true if both inputs have the same name and price
(define same-fruit?
  (lambda (F1 F2)
```

(f) (1 point) If you have two fruit structures that refer to the same fruit, you might want to combine them with a function like this:

```
;; merge-fruits:  fruit  fruit  ->  fruit
;; Return a single fruit structure with the name and price of the first input
;; and the other two fields formed by adding the corresponding fields of both inputs.
```

Suppose you have already defined this function; you don't have to define it here.

If `fruitA` describes apples for $1.50 a pound, with 350 pounds received and 250 pounds sold, and `fruitB` describes apples for $1.50 a pound, with 100 pounds received and 75 pounds sold, what is the value of this expression?

```
(fruit-received (merge-fruits fruitA fruitB))
```

(g) (4 points) Complete the definition of the function `merge-if-same` as described below. For full credit, use the function(s) described above where appropriate.

```
;; merge-if-same:  fruit  fruit  ->  fruit
;; If both inputs have the same name and price, return one fruit structure combining
;; the figures for both inputs.  Otherwise just return the first input unchanged.
(define merge-if-same
  (lambda (F1 F2)
```