Informatics 41 • Fall 2008 • David G. Kay & Rich Pattis • UC Irvine • Your name _____

SIXTH 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 (16 points)

We'll use `new-rrant` structures with menus, as in last week's lab, but we'll just call the structure `rrant` to save you some writing: `(define-struct rrant (name cuisine phone menu))` where the name, cuisine, and phone are strings and the menu is a list of dishes—`(define-struct dish (name price))`—where name is a string and price is a number.

On this quiz, we do not expect you to use `map`, `filter`, or `foldr`, but you *may* use them if you're confident enough to let your score depend on it. If it's appropriate in the definition of a function on this quiz to use a function previously defined on this quiz, we expect you to do that (for full credit) rather than duplicating the code defining the function.

(a) (4 points) Define the function `menu-contains?` as described below.

```
;; menu-contains?:  list-of-dish  string  ->  boolean
;; Return true if the string is the name of a dish on the list
(define menu-contains?
  (lambda (M s)
```

(b) (3 points) Define the function `rrant-serves?` as described below.

```
;; rrant-serves?:  rrant  string  ->  boolean
;; Return true if the rrant serves a dish with the given name
(define rrant-serves?
  (lambda (R s)
```

(c) (5 points) Define the function `rrants-serving` as described below.

```
;; rrants-serving:  list-of-rrant  string  ->  list-of-rrant
;; Return a list of the rrants in the input list that serve a dish with the given name
(define rrants-serving
  (lambda (L s)
```

(d) (4 points) Complete the definition of the function `average-price-of-dish` as described below; each blank should contain one constant or name. In your definition you may use the function `sum-prices-of-dish` as described below; you don't have to define it.

```
;; sum-prices-of-dish:  list-of-rrant  string  ->  number
;; The string names a dish; return sum of prices for that dish at all rrants on list

;; average-price-of-dish:  list-of-rrant  string  ->  number
;; The string names a dish; return the average price of that dish at the rrants
;; on the list that serve that dish.
(define average-price-of-dish
  (lambda (L s)
    (local ((define selected-rrants (_______________  _______________  _______________)))

      (_________________ (_________________ selected-rrants _________________)

                        (_________________  _________________)))))
```

Problem 2 (4 points)

Suppose we a binary tree with nodes defined as `(define-struct node (value left right))`, where the value field is a `rrant` as defined above. Complete the definition of `count-rrants-serving` below; each blank should contain one constant or name.

```
;; count-rrants-serving:  BT-of-rrant  string   ->  number
;; Return the number of rrants in the tree that serve the specified dish
(define count-rrants-serving
  (lambda (T s)
    (cond
      ((empty? T) _______________)

      (else (_______________

               (_______________ (_______________ T) s)

               (cond

                  ((_______________ (node-value T) s) 1)

                  (else 0))

               (_______________ (_______________ T) s))))))
```