Informatics 41 • Fall 2009 • David G. Kay • 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** (15 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)** (2 points) Define the function `dish-double-price` as described below.

```
;; dish-double-price:  dish  ->  dish
;; Return the input dish with its price doubled
(define dish-double-price
  (lambda (D)
```

**(b)** (4 points) Define the function `double-prices` as described below.

```
;; double-prices:  list-of-dish  ->  list-of-dish
;; Return the input list of dishes with each dish's price doubled
(define double-prices
  (lambda (L)
```

**(c)** (3 points) Define the function `rrant-num-dishes` as described below. (A predefined function will make this very easy; otherwise write a simple auxiliary function and call it from `rrant-num-dishes`.)

```
;; rrant-num-dishes:  rrant  ->  number
;; Return the number of dishes on the rrant's menu
(define rrant-num-dishes
  (lambda (R)
```

**(d)** (3 points) Define the function `total-dishes` as described below.

```
;; total-dishes:  list-of-rrant  ->  number
;; Return the total number of dishes served by all the rrants on the list
(define total-dishes
  (lambda (L)
```

**(e)** (3 points) Complete the definition of the function `average-dishes-per-rrant` as described below.

```
;; average-dishes-per-rrant:  list-of-rrant  ->  number
;; Return the average number of dishes served by restaurants on the list.
(define average-dishes-per-rrant
  (lambda (L)
```

**Problem 2** (5 points)

Suppose we a binary search tree with nodes defined as (define-struct node (value left right)), where the value field is a rrant defined as above. Assume that the following function is already defined:

```
;; rrant-serves?:  rrant  string  ->  boolean
;; Return true if the rrant's menu includes a dish whose name is the string
```

Complete the definition of `collect-rrants-serving` below; each blank should contain one constant or name.

```
;; collect-rrants-serving:  BST-of-rrant  string   ->  list-of-rrant
;; Return a list of all the rrants in the tree that serve the specified dish
(define collect-rrants-serving
  (lambda (T s)
    (cond
      ((empty? T) ________________)

      (else (________________

              (________________ (________________ T) s)

              (cond
                 ((________________ (node-value T) s) (list (________________ T)))

                 (else empty))

              (________________ (________________ T) s))))))
```