Informatics 41 • Fall 2006 • Kay/Thornton • UC Irvine Your name ________________________

FIFTH 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 (6 points)

A binary tree is either empty or `(make-node value left right)`, where value is a rrant and left and right are binary trees. Complete the definition of `sum-thai-prices` below. All the parentheses are in the correct places and each blank should be filled by exactly one item: symbol, function name, or constant. Restaurants are defined as usual:

```
(define-struct rrant (name cuisine phone dish price))
;; Thai?: rrant -> Boolean
;; Return true if the input restaurant serves Thai cuisine
(define Thai?
  (lambda (R)
    (string=? "Thai" (rrant-cuisine R))))
;; sum-thai-prices:  binary-tree-of-rrants -> number
;; Return the sum of the price fields of each Thai restaurant in the tree.
(define sum-thai-prices
  (lambda (T)
    (cond
      ((______________ T) ______________)

      (else (______________

            (______________ (______________ T))
            (cond
              ((______________ (______________ T)) (______________ (______________ T)))

              (else _______________))

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

Problem 2 (19 points)

Suppose we want to change the prices in a list of restaurants, but the amount to change each price depends on the cuisine of the restaurant. We could keep a list of cuisine-amount pairs like `(list '("Thai" 1.10) '("French" 2.00) '("Indian" 0.75))`, which says to increase the prices at Thai restaurants by 10%, double the prices at French restaurants, and decrease the prices at Indian restaurants by 25%. If we call each pair a modification, or "mod" for short, we can define a structure `(define-struct mod (cuisine amount))`.

(a) (6 points) Define the function `amount-for-cuisine` as described below.

```
;; amount-for-cuisine: string list-of-mod -> number
;; Return the number that matches the specified cuisine in the list of mods;
;; Return 1 if the string is not on list.
```

(b) (4 points) Our restaurant collection is a list of new-rrant structures as in last week's lab, `(define-struct new-rrant (name cuisine phone menu))`, where name, cuisine, and phone are strings and menu is a list of dish structures, `(define-struct dish (name price))`, where name is a string and price is a number.

Define the function `change-dish` as described below:

```
;; change-dish: dish number -> dish
;; Return the dish with the price changed by the specified amount (as described at
;; the top of this problem)
```

(c) (8 points) To change the prices in all the restaurants by an amount depending on the kind of cuisine served (as described above), define the function `change-all-by-cuisine` as described below. But before you start coding, read all of the following advice:

- Pay close attention to the types of the inputs and the types of the values each function returns.
- Use `amount-for-cuisine` described above (whether or not you wrote it correctly).
- Use this helper function; you do not have to define it:

  ```
  ;; change-one-rrant:  new-rrant number -> new-rrant
  ;; Return the input new-rrant with each price on its menu changed according
  ;; to the amount specified by the second input
  ```

- We do not expect you to use `map` or `filter` in your code on this quiz, but you may use them if you're confident enough to let your score depend on it.

```
;; change-all-by-cuisine: list-of-new-rrant list-of-mod -> list-of-new-rrant
;; Return the list of restaurants with each price changed according
;; to the change percentage for that restaurant's cuisine, as
;; specified on the mod list.
```

(d) (1 point) You wrote `change-dish`, but it didn't show up in any other code you wrote above. Where would it be used in this program?