

SIXTH 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 (8 points)

What is the value of each of the following expressions? (Remember that `even?` is a function that takes a number and returns true if it's evenly divisible by 2.)

```
(define add-some
  (lambda (test? L)
    (cond
      ((empty? L) 0)
      ((test? (first L)) (+ (first L) (add-some (rest L))))
      (else (add-some (rest L))))))
```

```
(define double
  (lambda (n)
    (* 2 n)))
```

- (a) `(add-some even? (list 1 3 2 4 7))`
- (b) `(add-some (lambda (n) (< n 10)) (list 1 23 4 2 17 49 1))`
- (c) `(filter even? (list 1 2 3 4 5 6 7 8))`
- (d) `(filter (lambda (n) (> (double n) 10)) (list 10 8 6 4 2))`
- (e) `(map double (list 1 3 5 7))`
- (f) `(map (lambda (n) (even? n)) (list 1 2 3 4 5))`
- (g) `(reduce 1 * (list 4 3 2 1))`
- (h) `(reduce 0 + (map double (list 1 2 3)))`
- (i) `(reduce empty append
 (list (map double (filter even? (list 4 3 2 1)))
 (map (lambda (n) (double (double n))) (list 1 2 3))
 (list 7)))`

Problem 2 (17 points)

Suppose we have a list called RL of restaurants with menus, according to the usual definitions:

```
(define-struct rrant (name cuisine phone menu))
```

where menu is a list of dishes

```
(define-struct dish (name price)).
```

(a) For each of the following expressions, describe in one English phrase what value it returns. Don't just say, "It does a reduce of zero and plus and ..."; give a description of what the expression *means*, something you could put in a software catalog so that a prospective buyer could find what he or she wanted.

(a.1) `(map rrant-name (filter Thai? RL))`

(a.2) `(map (lambda (R) (make-rrant (rrant-name R) (rrant-cuisine R) (rrant-phone R)
 (filter (lambda (D) (< (dish-price D) 10.00)) (rrant-menu R)))
 RL)`

(b) Using `map`, `filter`, and/or `reduce`, define each of the following functions without using explicit recursion.

```
;; north-african-restaurants: (listof rrant) -> (listof rrant)
```

```
;; Return a list of the restaurants in the input that serve either Moroccan, Tunisian, or
```

```
;; Ethiopian cuisine. (You may assume that Moroccan?, Tunisian?, and Ethiopian? are
```

```
;; already defined.)
```

```
;; menu-includes?: (listof dish) string -> boolean
```

```
;; Return true if string is the name of a dish on menu
```

Here's the full version of the last question; it wouldn't fit (in space or time) on the quiz. It's tough, but each of you should be able to work it out, especially if you break it down and tackle it step by step.

Using `map`, `filter`, and/or `reduce` and no explicit recursion, write an expression that returns a list of the names and phone numbers of all the restaurants that serve the dish "Bisteeya." Each name/phone-number pair should be in its own list; for example `((("Moun of Tunis" "343-3434") ("Koutoubia" "344-3334")))`.

Probably the only way to handle this is to define functions like these individually first:

```
;; menu-includes?: menu string -> boolean [This was on the quiz]
;; Return true if string is the name of a dish on menu
```

```
;; rrant-serves-dish?: rrant string ->boolean
;; Return true if string is the name of any menu item in rrant
```

```
;; make-name-phone: rrant -> list
;; Returns list containing rrant's name and phone
```