

## 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 (20 points)

Your electronic cookbook contains a list of recipes as defined in last week's lab assignment:

```
(define-struct recipe (title ingredients steps))
```

where the title is a symbol, ingredients is a list of symbols, and steps is a list of steps (where each step is a list of symbols); for example:

```
(make-recipe `ThaiIcedCoffee
  `(coffee sugar condensed-milk ice)
  `((brew coffee) (add sugar and condensed-milk) (pour coffee mixture over ice)))
```

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.

(a) (3 points) Define the function `recipe-contains-ingredient?` as described below; it's easy if you use the function `member?`.

```
;; recipe-contains-ingredient?: recipe symbol -> boolean
;; Return true if the symbol occurs in the recipe's ingredients list
(define recipe-contains-ingredient?
  (lambda (rec ing)
```

(b.1) (2 points) Define the function `step-short?` as described below; it's easy if you use `length`:

```
;; step-short?: list-of-symbols -> boolean
;; Take one recipe step (a list of symbols) as input;
;; return true if it contains 6 symbols (words) or fewer
(define step-short?
  (lambda (step)
```

**(b.2)** (5 points) Define the function `all-steps-short?` as described below. For full credit, use `step-short?` as described above (whether or not you defined it correctly).

```
;; all-steps-short?: list-of-steps -> boolean
;; Return true if each of the steps on the input list is short
;; (i.e., six words or fewer) or if list is empty
(define all-steps-short?
  (lambda (LOS)
```

**(b.3)** (4 points) Define the function `recipe-simple?` as described below. For full credit, use `all-steps-short?` where appropriate.

```
;; recipe-simple?: recipe -> boolean
;; Return true if recipe has fewer than 10 steps and each step is short (6 words or fewer)
(define recipe-simple?
  (lambda (rec)
```

**(b.4)** (6 points) The first word (symbol) of each step is a verb (e.g., `bake`, `mix`, `grill`); we can call that the *technique* involved in that step. Assume you have these functions; you don't have to define them:

```
;; step-involves-technique?: list-of-symbol symbol -> boolean
;; Return true if the first symbol on the list (the step)
;; matches the input symbol (the technique)

;; recipe-involves-technique?: recipe symbol -> boolean
;; Return true if any of the recipe's steps involves the technique (second argument).
```

Now, suppose you want to select some recipes that will help you practice a particular technique. Define the function `simple-practice-recipes` as described below:

```
;; simple-practice-recipes: list-of-recipe symbol -> list-of-recipe
;; Return all the recipes on the input list that are both simple and involve
;; the specified technique.
(define simple-practice-recipes
  (lambda (cookbook technique)
```