

How to Code: Simple Data


Module 4b: Reference Multiple Choice Quiz


For your convenience, this is a reference document for the Multiple Choice Quiz in Module 4b. Please note that not all quiz questions are included in this document, so be careful to answer every question in the actual quiz.


Questions 3-5

3 points possible (graded)

Consider the following partially obscured types comments:

```
;; ListA is one of:  
;; - empty  
;; - (cons "yes" ListA)  
;; - (cons "no" ListA)  
;; interp. 
```

```
;; ListB is one of:  
;; - empty  
;; (cons String ListB)  
;; interp. 
```

```
;; ListC is one of:  
;; - (cons String empty)  
;; - (cons String ListC)  
;; interp. 
```

The templates for these three types comments have also been partially obscured.

Template 1:

```
(define (fn-for-list l)
  (cond [(empty? (rest l)) (...)]
        [else
         (... (first l)
               (fn-for-list (rest l)))])])
```

Template 2:

```
(define (fn-for-list l)
  (cond [(empty? l) (...)]
        [(...) (... (first l)
                        (fn-for-list (rest l)))]
        [else
         (... (first l)
               (fn-for-list (rest l)))])])
```

Template 3:

```
(define (fn-for-list l)
  (cond [(empty? l) (...)]
        [else
         (... (first l)
               (fn-for-list (rest l)))])])
```

Question 10

1 point possible (graded)

Consider the following Data Definitions:

```
(define-struct concert (artist venue))
;; Concert is (make-concert String String)
;; interp. a concert with the band playing, and the venue they're
playing at
(define C1 (make-concert "Shakey Graves" "Commodore Ballroom"))
(define C2 (make-concert "Tallest Man On Earth" "Orpheum Theatre"))
#;
(define (fn-for-concert c)
  (... (concert-artist c)
        (fn-for-venue (concert-venue c))))

;; ListOfConcert is one of:
;; - empty
;; - (cons Concert ListOfConcert)
;; interp. a list of concerts
(define LOC1 empty)
(define LOC2 (cons C1 (cons C2 empty)))
#;
(define (fn-for-loc loc)
  (cond [(empty? loc)(...)]
        [else
         (... (fn-for-concert (first loc))
               (fn-for-loc (rest loc)))]))

(define-struct festival (name shows))
;; Festival is (make-festival String ListOfConcert)
;; interp. a festival with name, and list of shows that are part of
the festival
(define CANCELLED-FESTIVAL (make-festival "Cancelled" empty))
(define VFMF (make-festival "Vancouver Folk Music Festival"
                           (cons (make-concert "Hawksley Workman"
                                                "Main Stage")
                                (cons (make-concert "Grace Petrie"
                                                "Stage 1")
                                    (cons (make-concert "Mary
Gauthier" "Stage 5") empty)))))
#;
(define (fn-for-festival f)
  (... (festival-name f)
        (cond [(empty? (festival-shows f))(...)]
              [else
               (... (fn-for-concert (first (festival-shows f))
                                   (fn-for-festival (rest (festival-shows f)))))]))
```

Question 12

1 point possible (graded)

We want to design a function called `festival-schedule` that produces the lineup of a festival as a list of the bands and where they're performing as a list of strings.

So `(festival-schedule VFMF)` should produce

```
(cons "Hawksley Workman: Main Stage" (cons "Grace Petrie: Stage 1" (cons  
"Mary Gauthier: Stage 1" empty)))
```

```
;; Festival -> ListOfString  
;; produces a list of each band paired with where they are performing  
(check-expect (festival-schedule CANCELLED-FESTIVAL) empty)  
(check-expect (festival-schedule (make-festival "CFMF"  
                                                (cons (make-concert  
"Father John Misty" "Main Stage") empty)))  
              (cons "Father John Misty: Main Stage" empty))  
(define (festival-schedule f) f) ; stub
```

Question 13

1 point possible (graded)

Which of the following function definitions for `festival-schedule` **best** uses the reference rule?

```
(define (festival-schedule f)
  (cond [(empty? (festival-shows f)) empty]
        [else
         (cons (concert-helper (first (festival-shows f)))
               (festival-schedule (rest (festival-shows f)))))]))
```

```
(define (festival-schedule f)
  (cond [(empty? (festival-shows f)) empty]
        [else
         (cons (string-append (concert-artist (first (festival-shows
f))) ":")
               (concert-venue (first (festival-shows
f))))
         (festival-schedule (rest (festival-shows f)))))]))
```

```
(define (festival-schedule f)
  (display-all-concerts (festival-name f)
                        (festival-shows f)))

;; ListOfConcert -> ListOfString
;; produces a list of concert strings in format Artist Name: Venue
;; !!!
(define (display-all-concerts loc) empty)
```

```
(define (festival-schedule f)
  (display-all-concerts (festival-shows f)))

;; ListOfConcert -> ListOfString
;; produces a list of concert strings in format Artist Name: Venue
;; !!!
(define (display-all-concerts loc) empty)
```