Questions 1-7

Recall:

```
(define-struct person (name gender children))
;; Person is (make-person String Gender ListOfPerson)
;; interp. a person with first name, gender and a list of their children

;; ListOfPerson is one of:
;; - empty
;; (cons Person ListOfPerson)
;; interp. a list of persons

;; Gender is one of"
;; - "M"
;; - "F"
;; interp. "M" means male, "F" means female
```

Question 1

1/1 point (graded)

Which of the following is the correct template for `Person`?

○
```
(define (fn-for-person p)
  (... (person-name p)
       (person-gender p)
       (person-children p)))
```

○
```
(define (fn-for-person p)
  (... (fn-for-name (person-name p))
       (fn-for-gender (person-gender p))
       (fn-for-lop (person-children p))))
```

○
```
(define (fn-for-person p)
  (... (fn-for-person (person-name p))
       (fn-for-gender (person-gender p))
       (fn-for-lop (person-children p))))
```

●
```
(define (fn-for-person p)
  (... (person-name p)
       (fn-for-gender (person-gender p))
       (fn-for-lop (person-children p))))
```

✓

Explanation

Person is compound data, so you form the template using the selectors. But since the type of data produced by `(person-gender p)` and by `(person-children p)` are non-primitive, we wrap those in function calls to `fn-for-gender` and `fn-for-lop`.

Submit

Answers are displayed within the problem

Question 2

1/1 point (graded)

Which of the following is the correct template for `ListOfPerson`?

○

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

○

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

○

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

●

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

✓

Explanation

We design a template for self-referential data as usual, and add a helper function `fn-for-person` because `(first lop)` is of non-primitive type `Person`.

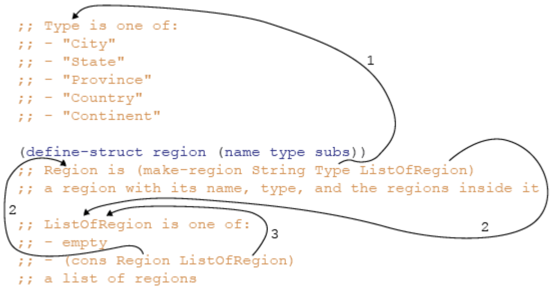
Submit

Answers are displayed within the problem

Question 3

1/1 point (graded)

In the following types comments, the reference arrows have been numbered, but the labels are missing:



```
;; Type is one of:
;; - "City"
;; - "State"
;; - "Province"
;; - "Country"
;; - "Continent"

(define-struct region (name type subs))
;; Region is (make-region String Type ListOfRegion)
;; a region with its name, type, and the regions inside it

;; ListOfRegion is one of:
;; - empty
;; - (cons Region ListOfRegion)
;; a list of regions
```

What is the correct correspondence between numbers and arrows??

☐ 1: SR

☑ 1: R

☐ 1: MR

☐ 2: SR

☐ 2: R

☑ 2: MR

☑ 3: SR

☐ 3: R

☐ 3: MR

✓

Explanation

There is a reference from Region to Type, a self reference within `ListOfRegion`, and a mutual reference cycle between Region and `ListOfRegion`.

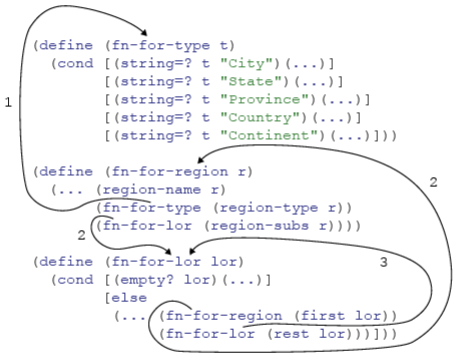
Submit

Answers are displayed within the problem

Question 4

1/1 point (graded)

Below are the templates for the above type comments. Select the correct number for each arrow to show the correspondence between the templates and the type comments



```
(define (fn-for-type t)
  (cond [(string=? t "City")(...)]
        [(string=? t "State")(...)]
        [(string=? t "Province")(...)]
        [(string=? t "Country")(...)]
        [(string=? t "Continent")(...)]))

(define (fn-for-region r)
  (... (region-name r)
       (fn-for-type (region-type r))
       (fn-for-lor (region-subs r))))

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

2	1	3	

✓

Submit

✓ Correct (1/1 point)

Question 5

1/1 point (graded)

In the templates, what should we label arrow(s) #1?

○ R

● NH

○ SR

○ NR

○ MR

○ NMR

✓

Explanation

If there is a reference in the type comment, there is a natual helper (NH) in the template.

Submit

Answers are displayed within the problem

Question 6

1/1 point (graded)

In the templates, what should we label arrow(s) #2?

○ R

○ NH

○ SR

○ NR

○ MR

● NMR

✓

Explanation

If there is a mutual reference cycle in the type comments, there is natual mutual recursion (NMR) in the templates.

Submit

Answers are displayed within the problem

Question 7

1/1 point (graded)

In the templates, what should we label arrow(s) #3?

○ R

○ NH

○ SR

● NR

○ MR

○ NMR

✓

Explanation

If there is a self-reference type comments, there is natual recursion (NR) in the templates.

Submit

Answers are displayed within the problem