Questions 2-4

Bookmark this page

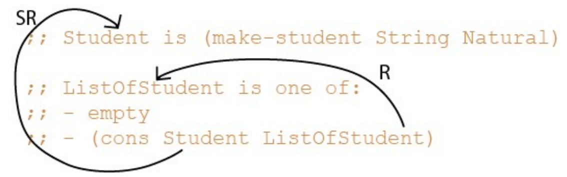
Question 2

1/1 point (graded)

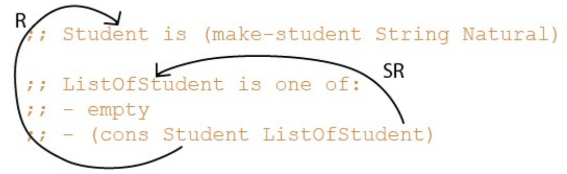
Consider the following type comments and interpretations:

```
(define-struct student (name id))
;; Student is (make-student String Natural)
;; interp. a student with name and student id

;; ListOfStudent is one of:
;; - empty
;; - (cons Student ListOfStudent)
;; interp. a list of students
```

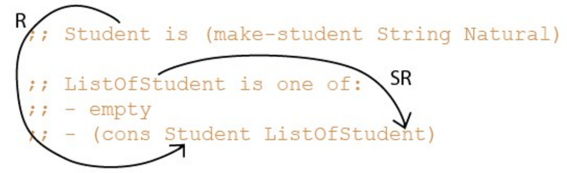
Which of the following is the correct way to draw the reference (R) and self-reference (SR) arrows?

○


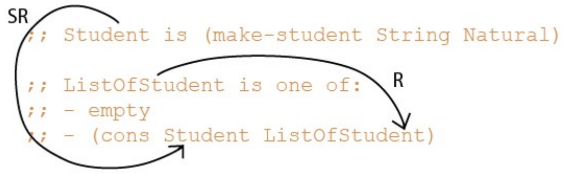
```
;; Student is (make-student String Natural)

;; ListOfStudent is one of:
;; - empty
;; - (cons Student ListOfStudent)
```

●


```
;; Student is (make-student String Natural)

;; ListOfStudent is one of:
;; - empty
;; - (cons Student ListOfStudent)
```

○


```
;; Student is (make-student String Natural)

;; ListOfStudent is one of:
;; - empty
;; - (cons Student ListOfStudent)
```

○


```
;; Student is (make-student String Natural)

;; ListOfStudent is one of:
;; - empty
;; - (cons Student ListOfStudent)
```

✓

Explanation
The self reference arrow should originate from the type reference, in this case `ListOfStudent` in `(cons Student ListOfStudent)`, and should point to the type name `ListOfStudent` (the word before "is").
The reference arrow should originate from the type reference, in this case `Student` in `(cons Student ListOfStudent)`, and should point to the type name it's referring to which is `Student` (the word before "is").

Submit

Show Answer

Answers are displayed within the problem

Question 3

1/1 point (graded)

Again consider the following data types:

```
(define-struct student (name id))
;; Student is (make-student String Natural)
;; interp. a student with name and student id

;; ListOfStudent is one of:
```

```
;; - empty
;; - (cons Student ListOfStudent)
;; interp. a list of students
```

The template for `Student` is:

```
#;
(define (fn-for-student s)
  (... (student-name s)
       (student-id s)))
```

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

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

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

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

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

✓

Explanation
Because the type of `(first los)` is Student, which is not a primitive type, we need to add a natural helper `fn-for-student`.

Submit

Show Answer

Answers are displayed within the problem

Question 4

1/1 point (graded)

Which template rule was used to add the call to `fn-for-student?` in the template for `ListOfStudent`?

● reference

○ self-reference

○ one-of

○ atomic distinct

✓

Explanation
Again, since the type of `(first los)` is Student, which is not a primitive type, the rule used to add the call to `fn-for-student` is the reference rule.

Submit

Show Answer

Answers are displayed within the problem

Previous Next