Next >

Design Recipes

Systematic Program Design - Part 2

Glossarv

Problem Bank

Course > 4b: Reference > The Reference Rule Part 1 > Questions 2-4

Language

 ✓ Previous

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

 □

Style Rules

Discussion

Progress

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?

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

;; ListOfStudent is one of:

;; - empty
; - (cons Student ListOfStudent)
```

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

;; ListOfStudent is one of: SR

;; - empty
; - (cons Student ListOfStudent)
```

```
R: Student is (make-student String Natural)

:: ListOfStudent is one of: SR

:: - empty
:- (cons Student ListOfStudent)
```

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

;; ListOfStudent is one of: R

;; - 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 <code>student</code> in (cons <code>student</code> ListOfStudent), and should point to the type name it's referring to which is <code>student</code> (the word before "is").

Submit

3 Show Answer

1 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 ${\tt student}$ is:

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

Which of the following is the correct template for ${\tt ListOfStudent?}$

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

```
(define (fn-for-los los)
  (cond [(empty? los)(...)]
           (... (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.



• Answers are displayed within the problem

Question 4

1/1 point (graded)

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

- reference o 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.



Answer

1 Answers are displayed within the problem

Previous Next >





edXAbout Legal

Connect

Blog









edX for Business

Honor Code Privacy Policy Accessibility Policy

Contact Us Help Center



© 2020 edX Inc. All rights reserved. | 深圳市恒宇博科技有限公司 <u>粤ICP备17044299</u> 号-2