Question 1

Bookmark this page

Question 1

1/1 point (graded)

You are asked to write a function that produces the number of rooms reachable from a given room, including that room itself. You decide that you need an additional result-so-far accumulator to complete this function, and add it to the template like this (tests have been omitted to save space):

```
;; Room -> Natural
;; produce the total number of rooms reachable from a given room, including the room itself

(define (num-rooms r0)
  ;; todo is (listof Room); a worklist accumulator
  ;; visited is (listof String); context preserving acc, names of rooms already visited
  (local [(define (fn-for-room r todo visited rsf)
            (if (member (room-name r) visited)
                (fn-for-lor todo visited (... rsf))
                (fn-for-lor (append (room-exits r) todo)
                            (cons (room-name r) visited)
                            (... rsf))))
          (define (fn-for-lor todo visited rsf)
            (cond [(empty? todo) (... rsf)]
                  [else
                   (fn-for-room (first todo)
                                (rest todo)
                                visited
                                (... rsf))]))]
    (fn-for-room r0 empty empty)))
```

Is anything wrong with or missing from the new template?

- [ ] Nope! Looks good
- [ ] We need to add `rsf` after `r0` in the first line
- [x] We need to initialize `rsf` in the trampoline
- [ ] We only need rsf in `fn-for-room` not `fn-for-lor`
- [x] We need a comment about the type and invariant of `rsf`

✔

Explanation

We can set the initial value of `rsf` as 0, and add a comment about the type and invariant as follows:

```
;; rsf is Natural; number of rooms visited so far
```

Submit Show Answer

Answers are displayed within the problem