Question 4

Bookmark this page

Question 4

1/1 point (graded)

Now we have:

```
;; ListOfString -> ListOfString
;; sort strings into alphabetical order
(check-expect (sort-strings empty) empty)
(check-expect (sort-strings (cons S1 (cons S2 empty))) (cons S1 (cons S2 empty)))
(check-expect (sort-strings (cons S3 (cons S1 empty))) (cons S1 (cons S3 empty)))

; (define (sort-strings los) los)
```

Choose the correct function body for `sort-strings`. Assume we wished for a function with this wish list entry:

```
;; String ListOfString -> ListOfString
;; insert s in the correct place in the sorted list los
(define (insert-string s los) los)
```

```
(define (sort-strings los)
  (cond [(empty? los) empty]
        [else
         (insert-string (first los)
                        (sort-strings (rest los)))]))
```

```
(define (sort-strings los)
  (cond [(empty? los) empty]
        [else
         (sort-strings (first los)
                       (insert-string (rest los)))]))
```

```
(define (sort-strings los)
  (cond [(empty? los) empty]
        [else
         (insert-string (first los)
                        (rest los))]))
```

Explanation

Like `sort-images`, all we need to do in `sort-strings` is to insert the string in the right place in a sorted list. So we need to trust that the result of the natural recursion is going to be a sorted list, and wish for a helper that does the insertion.

Submit Show Answer

Answers are displayed within the problem