Concurso Auxiliar de 2da categoria

Programacion I

Damian Ariel Marotte

Diseñe la funcion `cortas` que tome una *una lista de strings* y sin usar devuelva una *lista* con aquellas palabras de longitud menor a 5.

Ejemplo:

```
(cortas (list "Lista" "de" "palabras" "sin" "sentido")) == (list "de" "sin")
```

Receta

1. Diseño de datos
2. Signatura
3. Declaracion de proposito
4. Ejemplos
5. Codigo
6. Testing
7. Correccion

Diseño de datos

Una *ListaDeStrings* es:

- `Una lista vacia '()`
- `Una expresion del tipo (cons String ListaDeStrings)`

Una *ListaDeStrings* es:

- `Una lista vacia '()`
- `Una expresion del tipo (cons String ListaDeStrings)`

Predicados:

- `empty?` `; Reconoce unicamente la lista vacia`
- `cons?` `; Reconoce listas no vacias`

Una *ListaDeStrings* es:

- `Una lista vacia '()`
- `Una expresion del tipo (cons String ListaDeStrings)`

Predicados:

- `empty?` `; Reconoce unicamente la lista vacia`
- `cons?` `; Reconoce listas no vacias`

Selectores:

- `first` `; Devuelve el primer elemento de una lista`
- `rest` `; Devuelve la lista sin su primer elemento`

Una *ListaDeStrings* es:

- `Una lista vacia '()`
- `Una expresion del tipo (cons String ListaDeStrings)`

Predicados:

- `empty?` *`; Reconoce unicamente la lista vacia`*
- `cons?` *`; Reconoce listas no vacias`*

Selectores:

- `first` *`; Devuelve el primer elemento de una lista`*
- `rest` *`; Devuelve la lista sin su primer elemento`*

Funciones de strings:

- `string-length` *`; Devuelve la cantidad de caracteres de un string`*

Diseño de datos

```
;l: ListaDeStrings (Representa la lista de cadenas a filtrar)
```

Signatura

```
;l: ListaDeStrings (Representa la lista de cadenas a filtrar)
;cortas : ListaDeStrings -> ListaDeStrings
```

Declaracion de proposito

```
;l: ListaDeStrings (Representa la lista de cadenas a filtrar)
;cortas : ListaDeStrings -> ListaDeStrings
;Dada una lista de strings, devuelve una lista con aquellas palabras de largo menor a 5.
```

Ejemplos

```
1  ;l: ListaDeStrings (Representa la lista de cadenas a filtrar)
2  ;cortas : ListaDeStrings -> ListaDeStrings
3  ;Dada una lista de strings, devuelve una lista con aquellas palabras de largo menor a 5.
4
5
6
7
8
9
10
11
12
13 ;(cortas '())                                              == '()
14 ;(cortas (list "Palabras" "largas"))                       == '()
15 ;(cortas (list "Yo" "soy" "asi"))                          == (list "Yo" "soy" "asi")
16 ;(cortas (list "Lista" "de" "palabras" "sin" "sentido")) == (list "de" "sin")
17
18
19
20
```

Codigo

```
1  ;l: ListaDeStrings (Representa la lista de cadenas a filtrar)
2  ;cortas : ListaDeStrings -> ListaDeStrings
3  ;Dada una lista de strings, devuelve una lista con aquellas palabras de largo menor a 5.
4
5  (define (cortas l)
6
7
8
9
10
11 )
12
13 ;(cortas '())                                                  == '()
14 ;(cortas (list "Palabras" "largas"))                          == '()
15 ;(cortas (list "Yo" "soy" "asi"))                            == (list "Yo" "soy" "asi")
16 ;(cortas (list "Lista" "de" "palabras" "sin" "sentido")) == (list "de" "sin")
17
18
19
20
```

Codigo

```
1  ;l: ListaDeStrings (Representa la lista de cadenas a filtrar)
2  ;cortas : ListaDeStrings -> ListaDeStrings
3  ;Dada una lista de strings, devuelve una lista con aquellas palabras de largo menor a 5.
4
5  (define (cortas l)
6      (cond [(empty? l)    ]                                    ; Caso base
7            [(cons? l)                                         ; Caso recursivo
8
9                                                  ]
10     )
11 )
12
13 ;(cortas '())                                          == '()
14 ;(cortas (list "Palabras" "largas"))                   == '()
15 ;(cortas (list "Yo" "soy" "asi"))                      == (list "Yo" "soy" "asi")
16 ;(cortas (list "Lista" "de" "palabras" "sin" "sentido")) == (list "de" "sin")
17
18
19
20
```

Codigo

```
1  ;l: ListaDeStrings (Representa la lista de cadenas a filtrar)
2  ;cortas : ListaDeStrings -> ListaDeStrings
3  ;Dada una lista de strings, devuelve una lista con aquellas palabras de largo menor a 5.
4
5  (define (cortas l)
6      (cond [(empty? l) '()]                                    ; Caso base
7            [(cons? l)                                          ; Caso recursivo
8
9                                                 ]
10     )
11 )
12
13 ;(cortas '())                                          == '()
14 ;(cortas (list "Palabras" "largas"))                   == '()
15 ;(cortas (list "Yo" "soy" "asi"))                      == (list "Yo" "soy" "asi")
16 ;(cortas (list "Lista" "de" "palabras" "sin" "sentido")) == (list "de" "sin")
17
18
19
20
```

Codigo

```
1  ;l: ListaDeStrings (Representa la lista de cadenas a filtrar)
2  ;cortas : ListaDeStrings -> ListaDeStrings
3  ;Dada una lista de strings, devuelve una lista con aquellas palabras de largo menor a 5.
4
5  (define (cortas l)
6      (cond [(empty? l) '()]                                          ; Caso base
7            [(cons? l)  (if (< (string-length (first l)) 5)           ; Condicion
8                            (                                  ) ; Caso verdadero
9                            (                ))]                  ; Caso falso
10     )
11 )
12
13 ;(cortas '())                                          == '()
14 ;(cortas (list "Palabras" "largas"))                   == '()
15 ;(cortas (list "Yo" "soy" "asi"))                      == (list "Yo" "soy" "asi")
16 ;(cortas (list "Lista" "de" "palabras" "sin" "sentido")) == (list "de" "sin")
17
18
19
20
```

Codigo

```
1  ;l: ListaDeStrings (Representa la lista de cadenas a filtrar)
2  ;cortas : ListaDeStrings -> ListaDeStrings
3  ;Dada una lista de strings, devuelve una lista con aquellas palabras de largo menor a 5.
4
5  (define (cortas l)
6      (cond [(empty? l) '()]                                    ; Caso base
7            [(cons? l)  (if (< (string-length (first l)) 5)     ; Condicion
8                           (cons (first l)                  ) ; Caso verdadero
9                           (               ))]                ; Caso falso
10     )
11 )
12
13 ;(cortas '())                                          == '()
14 ;(cortas (list "Palabras" "largas"))                   == '()
15 ;(cortas (list "Yo" "soy" "asi"))                      == (list "Yo" "soy" "asi")
16 ;(cortas (list "Lista" "de" "palabras" "sin" "sentido")) == (list "de" "sin")
17
18
19
20
```

Codigo

```
1  ;l: ListaDeStrings (Representa la lista de cadenas a filtrar)
2  ;cortas : ListaDeStrings -> ListaDeStrings
3  ;Dada una lista de strings, devuelve una lista con aquellas palabras de largo menor a 5.
4
5  (define (cortas l)
6      (cond [(empty? l) '()]                                    ; Caso base
7            [(cons? l)  (if (< (string-length (first l)) 5)     ; Condicion
8                            (cons (first l) (cortas (rest l))) ; Caso verdadero
9                            (                ))]               ; Caso falso
10     )
11 )
12
13 ;(cortas '())                                                 == '()
14 ;(cortas (list "Palabras" "largas"))                         == '()
15 ;(cortas (list "Yo" "soy" "asi"))                           == (list "Yo" "soy" "asi")
16 ;(cortas (list "Lista" "de" "palabras" "sin" "sentido")) == (list "de" "sin")
17
18
19
20
```

Codigo

```
1  ;l: ListaDeStrings (Representa la lista de cadenas a filtrar)
2  ;cortas : ListaDeStrings -> ListaDeStrings
3  ;Dada una lista de strings, devuelve una lista con aquellas palabras de largo menor a 5.
4
5  (define (cortas l)
6      (cond [(empty? l) '()]                                    ; Caso base
7            [(cons? l)  (if (< (string-length (first l)) 5)     ; Condicion
8                            (cons (first l) (cortas (rest l))) ; Caso verdadero
9                            (cortas (rest l)))]                ; Caso falso
10     )
11 )
12
13 ;(cortas '())                                                 == '()
14 ;(cortas (list "Palabras" "largas"))                         == '()
15 ;(cortas (list "Yo" "soy" "asi"))                            == (list "Yo" "soy" "asi")
16 ;(cortas (list "Lista" "de" "palabras" "sin" "sentido")) == (list "de" "sin")
17
18
19
20
```

Testing

```
1  ;l: ListaDeStrings (Representa la lista de cadenas a filtrar)
2  ;cortas : ListaDeStrings -> ListaDeStrings
3  ;Dada una lista de strings, devuelve una lista con aquellas palabras de largo menor a 5.
4
5  (define (cortas l)
6      (cond [(empty? l) '()]                                    ; Caso base
7            [(cons? l)  (if (< (string-length (first l)) 5)     ; Condicion
8                            (cons (first l) (cortas (rest l))) ; Caso verdadero
9                            (cortas (rest l)))]                ; Caso falso
10     )
11 )
12
13 ;(cortas '())                                                 == '()
14 ;(cortas (list "Palabras" "largas"))                         == '()
15 ;(cortas (list "Yo" "soy" "asi"))                            == (list "Yo" "soy" "asi")
16 ;(cortas (list "Lista" "de" "palabras" "sin" "sentido")) == (list "de" "sin")
17 (check-expect (cortas '()) '())
18 (check-expect (cortas (list "Palabras" "largas")) '())
19 (check-expect (cortas (list "Yo" "soy" "asi")) (list "Yo" "soy" "asi"))
20 (check-expect (cortas (list "Lista" "de" "palabras" "sin" "sentido")) (list "de" "sim"))
```

Ran 4 tests.
1 of the 4 tests failed.

No signature violations.

Check failures:
Actual value **(list “de” “sin”)** differs from **(list “de” “sim”)**, the expected value.
at line 20, column 0

Correccion

```
1  ;l: ListaDeStrings (Representa la lista de cadenas a filtrar)
2  ;cortas : ListaDeStrings -> ListaDeStrings
3  ;Dada una lista de strings, devuelve una lista con aquellas palabras de largo menor a 5.
4
5  (define (cortas l)
6      (cond [(empty? l) '()]                                    ; Caso base
7            [(cons? l)  (if (< (string-length (first l)) 5)     ; Condicion
8                            (cons (first l) (cortas (rest l))) ; Caso verdadero
9                            (cortas (rest l)))]                ; Caso falso
10     )
11 )
12
13 ;(cortas '())                                                 == '()
14 ;(cortas (list "Palabras" "largas"))                         == '()
15 ;(cortas (list "Yo" "soy" "asi"))                           == (list "Yo" "soy" "asi")
16 ;(cortas (list "Lista" "de" "palabras" "sin" "sentido")) == (list "de" "sin")
17 (check-expect (cortas '()) '())
18 (check-expect (cortas (list "Palabras" "largas")) '())
19 (check-expect (cortas (list "Yo" "soy" "asi")) (list "Yo" "soy" "asi"))
20 (check-expect (cortas (list "Lista" "de" "palabras" "sin" "sentido")) (list "de" "sim"))
```

Correccion

```
;l: ListaDeStrings (Representa la lista de cadenas a filtrar)
;cortas : ListaDeStrings -> ListaDeStrings
;Dada una lista de strings, devuelve una lista con aquellas palabras de largo menor a 5.

(define (cortas l)
    (cond [(empty? l) '()]                                    ; Caso base
          [(cons? l)  (if (< (string-length (first l)) 5)     ; Condicion
                          (cons (first l) (cortas (rest l))) ; Caso verdadero
                          (cortas (rest l)))]                ; Caso falso
    )
)

;(cortas '())                                                 == '()
;(cortas (list "Palabras" "largas"))                          == '()
;(cortas (list "Yo" "soy" "asi"))                             == (list "Yo" "soy" "asi")
;(cortas (list "Lista" "de" "palabras" "sin" "sentido")) == (list "de" "sin")
(check-expect (cortas '()) '())
(check-expect (cortas (list "Palabras" "largas")) '())
(check-expect (cortas (list "Yo" "soy" "asi")) (list "Yo" "soy" "asi"))
(check-expect (cortas (list "Lista" "de" "palabras" "sin" "sentido")) (list "de" "sin"))
```

All 4 test passed!

Codigo

```
1  ;l: ListaDeStrings (Representa la lista de cadenas a filtrar)
2  ;cortas : ListaDeStrings -> ListaDeStrings
3  ;Dada una lista de strings, devuelve una lista con aquellas palabras de largo menor a 5.
4
5  (define (cortas l)
6      (cond [(empty? l) '()]                                    ; Caso base
7            [(cons? l)  (if (< (string-length (first l)) 5)     ; Condicion
8                            (cons (first l) (cortas (rest l))) ; Caso verdadero
9                            (cortas (rest l)))]                ; Caso falso
10     )
11 )
12
13 ;(cortas '())                                                 == '()
14 ;(cortas (list "Palabras" "largas"))                         == '()
15 ;(cortas (list "Yo" "soy" "asi"))                            == (list "Yo" "soy" "asi")
16 ;(cortas (list "Lista" "de" "palabras" "sin" "sentido")) == (list "de" "sin")
17 (check-expect (cortas '()) '())
18 (check-expect (cortas (list "Palabras" "largas")) '())
19 (check-expect (cortas (list "Yo" "soy" "asi")) (list "Yo" "soy" "asi"))
20 (check-expect (cortas (list "Lista" "de" "palabras" "sin" "sentido")) (list "de" "sin"))
```