

# Concurso Auxiliar de 2da categoria

Programacion I

Damian Ariel Marotte

## Practica 5: Primera parte

**Ejercicio 13** 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")
```

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

Una *ListaDeStrings* es:

- Una lista vacía `'()`
- Una expresión 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*

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

```
2
```

```
3
```

```
4
```

```
5
```

```
6
```

```
7
```

```
8
```

```
9
```

```
10
```

```
11
```

```
12
```

```
13
```

```
14
```

```
15
```

```
16
```

```
17
```

```
18
```

```
19
```

```
20
```



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

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

# Declaracion de proposito

```
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
14
15
16
17
18
19
20
```

# 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
```

```
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
```

```
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
```

```

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                             (
9                                 (
10                                     ))] ; Caso falso
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

```

```

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                                 (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

```



```
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
```

```
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" "sin"))
```

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*

```
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"))
```

```
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"))
```

*All 4 test passed!*

```

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"))

```