

2.2) 6) Esta función es necesaria para las demás

```
(define (reverse lista l-reverse)

  (cond

    ((null? lista) l-reverse)

    ((list? (car lista)) (reverse (cdr lista) (cons (reverse(car lista) '()) l-reverse)))

    (else (reverse (cdr lista) (cons (car lista) l-reverse))))

  )

)
```

```
(define (deep-reverse lista)

  (cond

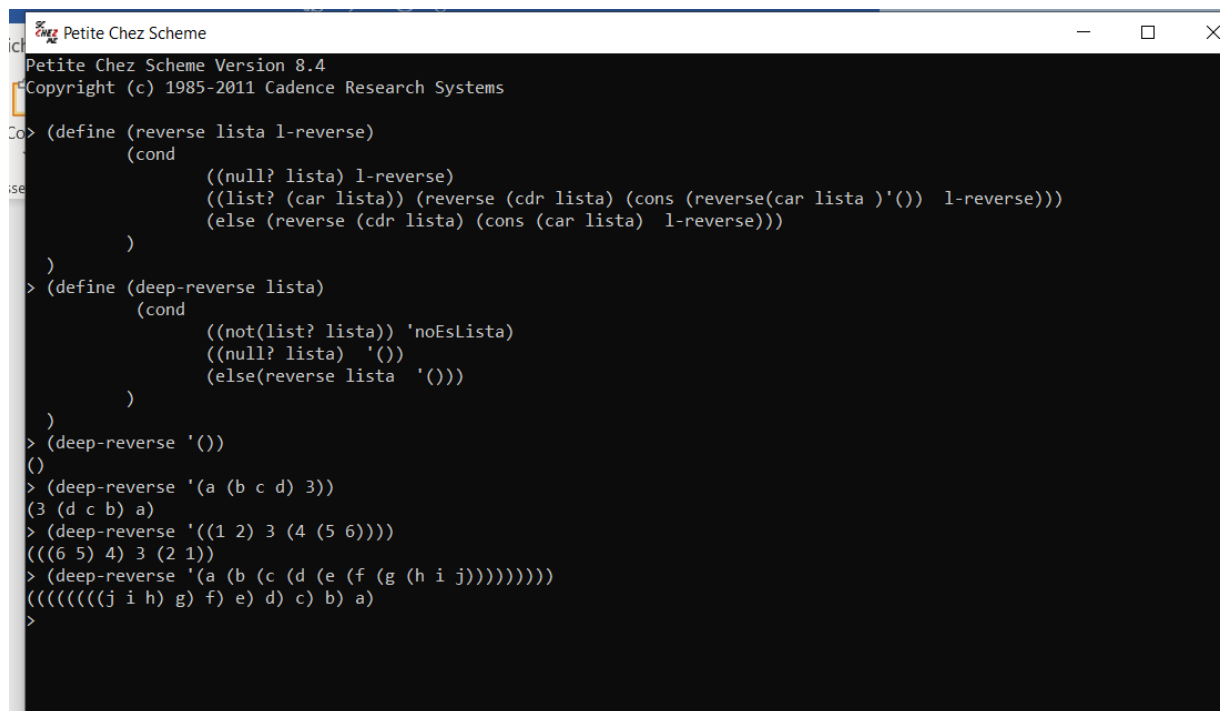
    ((not(list? lista)) 'noEsLista)

    ((null? lista) '())

    (else(reverse lista '())))

  )

)
```



```
Petite Chez Scheme
Petite Chez Scheme Version 8.4
Copyright (c) 1985-2011 Cadence Research Systems

Co> (define (reverse lista l-reverse)
      (cond
        ((null? lista) l-reverse)
        ((list? (car lista)) (reverse (cdr lista) (cons (reverse(car lista) '()) l-reverse)))
        (else (reverse (cdr lista) (cons (car lista) l-reverse))))
      )
    )
> (define (deep-reverse lista)
      (cond
        ((not(list? lista)) 'noEsLista)
        ((null? lista) '())
        (else(reverse lista '())))
      )
    )
> (deep-reverse '())
()
> (deep-reverse '(a (b c d) 3))
(3 (d c b) a)
> (deep-reverse '((1 2) 3 (4 (5 6))))
(((6 5) 4) 3 (2 1))
> (deep-reverse '(a (b (c (d (e (f (g (h i j))))))))))
((((((((j i h) g) f) e) d) c) b) a)
>
```

2.1) 6)

PARA DAR LA VUELTA LA LISTA

```
(define (reverse lista l-reverse)
```

```
  (cond
```

```
    ((null? lista) l-reverse)
```

```
    ((list? (car lista)) (reverse (cdr lista) (cons (reverse(car lista) '()) l-reverse)))
```

```
    (else (reverse (cdr lista) (cons (car lista) l-reverse))))
```

```
  )
```

```
)
```

```
(define (deep-reverse lista)
```

```
  (cond
```

```
    ((not(list? lista)) 'noEsLista)
```

```
    ((null? lista) '())
```

```
    (else(reverse lista '()))
```

```
  )
```

```
)
```

```
(define (duplicate-l lista l-duplicada)
```

```
  (cond
```

```
    ((null? lista) (deep-reverse l-duplicada))
```

```
    (else (duplicate-l (cdr lista) (cons (car lista) (cons (car lista) l-duplicada) )))
```

```

    )
)

(define (duplicate lista)
  (cond
    ((not(list? lista)) 'noEsLista)
    ((null? lista) '())
    (else(duplicate-l lista '())))
  )
)

```

```

Petite Chez Scheme
> (deep-reverse '())
()
> (deep-reverse '(a (b c d) 3))
(3 (d c b) a)
> (deep-reverse '((1 2) 3 (4 (5 6))))
(((6 5) 4) 3 (2 1))
> (deep-reverse '(a (b (c (d (e (f (g (h i j))))))))
((((((((j i h) g) f) e) d) c) b) a)
>
  (define (duplicate-l lista l-duplicada)
    (cond
      ((null? lista) (deep-reverse l-duplicada))
      (else (duplicate-l (cdr lista) (cons (car lista) (cons (car lista) l-duplicada) ))))
    )
  )
> (define (duplicate lista)
  (cond
    ((not(list? lista)) 'noEsLista)
    ((null? lista) '())
    (else(duplicate-l lista '())))
  )
)
> (duplicate '(1 2 3 4 5))
(1 1 2 2 3 3 4 4 5 5)
> (duplicate '())
()
> (duplicate '(a b c d e f g h))
(a a b b c c d d e e f f g g h h)
>

```

10)

FUNCION PARA DAR LA VUELTA LA LISTA

```
(define (reverse lista l-reverse)
```

```
  (cond
```

```

        ((null? lista) l-reverse)
        ((list? (car lista)) (reverse (cdr lista) (cons (reverse(car lista) '()) l-reverse)))
        (else (reverse (cdr lista) (cons (car lista) l-reverse)))
    )
)

```

```

(define (deep-reverse lista)
    (cond
        ((not(list? lista)) 'noEsLista)
        ((null? lista) '())
        (else(reverse lista '())))
    )
)

```

```

(define (positives-l lista l-positiva)
    (cond
        ((null? lista) (deep-reverse l-positiva))
        ((> (car lista) 0) (positives-l (cdr lista) (cons (car lista) l-positiva)))
        ((> 0 (car lista)) (positives-l (cdr lista) l-positiva))
    )
)

```

```

(define (positives lista)
    (cond
        ((not(list? lista)) 'noEsLista)
        ((null? lista) '())
        (else(positives-l lista '())))
    )
)

```

)
)

```
)  
)  
> (define (deep-reverse lista)  
  (cond  
    ((not(list? lista)) 'noEsLista)  
    ((null? lista) '())  
    (else(reverse lista '())))  
  )  
)  
> (define (positives-l lista l-positiva)  
  (cond  
    ((null? lista) (deep-reverse l-positiva))  
    ((> (car lista) 0) (positives-l (cdr lista) (cons (car lista) l-positiva)))  
    ((> 0 (car lista)) (positives-l (cdr lista) l-positiva))  
  )  
)  
> (define (positives lista)  
  (cond  
    ((not(list? lista)) 'noEsLista)  
    ((null? lista) '())  
    (else(positives-l lista '()))  
  )  
)  
> (positives '(-4 -1 -10 -13 -5))  
(  
> (positives '(12 -4 3 -1 -10 -13 6 -5))  
(12 3 6)  
> (positives '())  
(  
>
```