004. Hallar el enésimo elemento de una lista

Escriba una función que dados un número natural mayor a cero y una lista de elementos, retorne el elemento que ocupa la posición indicada por el número. Si la lista no contiene suficientes elementos, debe producir un error indicando el problema.

Para comenzar a resolverlo, simplificamos el enunciado, y vamos a suponer que si se solicita el elemento de la posición 3, digamos, la lista tiene al menos 3 elementos. Ejemplo:

```
(n-esimo 3 ‘(a b c d)) -> ‘c
```

En este caso se solicita el elemento que ocupa la posición 3, y la lista dispone de cuatro elementos en total.

¿Cuál es la posición trivial a la hora de conseguir un elemento? Es claro que la 1, que corresponde a la cabeza de la lista, por ejemplo:

```
(n-esimo 1 ‘(a b c d)) -> ‘a
```

¿Que pasa si busco el tercer elemento de la lista?

```
(n-esimo 3 ‘(a b c d))
```

Sabemos que el 3 no es el 1, que es el caso inmediato. Sabemos que 3 está después de de la cabeza. Podemos descartar la cabeza y transformar la búsqueda del tercer elemento de la lista ‘(a b c d), en la búsqueda del segundo elemento de ‘(b c d)

```
(n-esimo 2 ‘(b c d))
```

y nuevamente:

```
(n-esimo 1 ‘(c d))
```

el cual tiene respuesta inmediata: **‘c**

Codificamos eso en Scheme:

```
(define n-esimo
  (lambda (n lst)
    (if (= n 1)
      (car lst)
      (n-esimo (- n 1) (cdr lst)))))
```

¿Qué pasa con nuestro código cuando lo invocamos con una lista muy corta?

```
(n-esimo 8 '(a b c d))
```

Cada llamada recursiva reduce en uno el **n**, y reduce en un elemento la lista **lst**. Cuando **lst** sea vacía, se va a intentar reducir una vez más su longitud con **cdr** y va a dar un error, pues **cdr** no está definido para listas vacías.

Completemos el enunciado para detectar esta situación y mostrar un mensaje de error apropiado.

Si **lst** está vacía al entrar en la función, se debe mostrar un error. Codifiquemos eso, insertándolo antes del resto del código.

```
(define n-esimo
  (lambda (n lst)
    (if (null? lst) (error "n-esimo: no hay suficientes elementos")
        (if (= n 1)
          (car lst)
          (n-esimo (- n 1) (cdr lst))))))
```

La función **error** no está presente en R5RS. Es una función que tiene una larga trayectoria en la familia de las diversas implementaciones Scheme. Desde 2001, en SRFI-23 (*Scheme Request For Implementation*[1]) se dan las pautas para la implementación de **error** como *procedure*:

https://srfi.schemers.org/srfi-23/srfi-23.html

En R6RS ya está disponible oficialmente:
http://www.r6rs.org/final/html/r6rs/r6rs-Z-H-14.html#node_sec_11.14

El estándar R6RS tiene un elaborado mecanismo de tratamiento de excepciones, que el lector avanzado puede leer en:
http://www.r6rs.org/final/html/r6rs-lib/r6rs-lib-Z-H-8.html#node_chap_7

[1] https://srfi.schemers.org/ Scheme Requests for Implementation