**TC2006 – Lenguajes de Programación***Funciones de Orden Superior en Scheme*

Matrícula1\_\_\_\_\_\_\_\_\_\_\_ Matrícula2 \_\_\_\_\_\_\_\_\_\_\_\_

1. Programar la función recursiva **nprocs** en Scheme que cuente el número de procedimientos NO primitivos dentro de una lista posiblemente imbricada.

Probar con:

> (define (sqr x) (\* x x))

> (define inc1 (lambda (x) (+ x 1)))

> (nprocs ‘(+ (sqr 1) (\* 2 (inc1 3)))) => 0

> (nprocs (+ (sqr 1) (\* 2 (inc1 3)))) => ERROR

> (nprocs (list + (list sqr 1) (list \* 2 (list inc1 3)))) => 2

1. Programar la función recursiva **nveces** en Scheme que aplique N veces un procedimiento a un valor.

Probar con:

> (nveces 3 (lambda (x) (\* x x)) 2) => 256

> (nveces 2 cdr ‘(1 2 3)) => (3)

1. Programar la función recursiva **mapea** en Scheme que aplique una función que reciba como argumento a cada uno de los elementos de una lista.

Probar con:

> (mapea (lambda (x) (\* x x)) ‘(1 2 3)) => (1 4 9)

> (mapea cdr ‘((1 2)(3 4))) => ((2)(4))

1. Programar la función recursiva **aplica-n** en Scheme que sirva para generar procedimientos que apliquen N veces sobre un valor, la función dada como parámetro de entrada.

Probar con:

> (define inc7 (aplica-n (lambda (x) (+ x 1)) 7))

> (inc7 5) => 12

> (define eleva8a (aplica-n (lambda (x) (\* x x)) 3)

> (eleva8a 4) => 64

> (define quita4 (aplica-n cdr 4))

> (quita4 '(1 2 3 4 5)) => (5)