PRACTICA 01 – COMMON LISP

```
;;; EJERCICIO 1
;;; Definir una función que, dados la base y la altura de un
;;; triángulo, calcule su área.

;;; Ejemplos:
; (área-triángulo 3 3) ==> 9/2
; (área-triángulo 1.4 5.8) ==> 4.06


;;; EJERCICIO 2
;;; Definir una función que, dados un número real y, opcionalmente, un
;;; número de cifras decimales con valor por defecto 0, redondee el número
;;; real a ese número de cifras decimales.

;;; Ejemplos:
; (redondear 3.14159265) ==> 3.0
; (redondear 3.14159265 4) ==> 3.1416

;;; Definir una función que, dado un valor en una cierta unidad de
;;; medida, lo transforme a otra unidad de medida, estableciéndose
;;; mediante una clave numérica la equivalencia entre las dos unidades
;;; y mediante otra clave lógica si la transformación es directa (por
;;; defecto) o inversa.

;;; Ejemplos, sabiendo que 1 libra = 0.45359237 kilogramos y
;;; 1 kilogramo = 2.20462262 libras:
; (convertir-unidades 1 :equivalencia 0.45359237) ==> 0.45359
; (convertir-unidades 1 :equivalencia 0.45359237 :inversa t) ==> 2.20462
; (convertir-unidades 2 :equivalencia 2.20462262) ==> 4.40924


;;; EJERCICIO 3
;;; Definir una función que, dados los coeficientes de una ecuación de
;;; segundo grado, devuelva el número de soluciones reales distintas
;;; que posee la ecuación.

;;; Ejemplos:
; (contar-soluciones-ecuación 1 -3 2) ==> 2
; (contar-soluciones-ecuación 1 -2 1) ==> 1
; (contar-soluciones-ecuación 1 1 1) ==> 0


;;; EJERCICIO 4
;;; Un año es bisiesto si es divisible por 4 y no lo es por 100, a
;;; menos que lo sea por 400.
;;; Definir un predicado que determine, dado un año, si es o no
;;; bisiesto.
```

```
;;; Ejemplos:
; (año-bisiesto-p 2001) ==> NIL
; (año-bisiesto-p 2004) ==> T
; (año-bisiesto-p 2100) ==> NIL
; (año-bisiesto-p 2000) ==> T


;;; EJERCICIO 5
;;; Consideremos la sucesión de números enteros positivos definida por
;;; recursión como sigue:

;;;              /
;;;              | 3a_n + 1, si a_n es impar
;;;          /
;;; a_(n+1) = \  a_n
;;;              | ---,      si a_n es par
;;;              \  2

;;;; Definir una función que, dado un número entero positivo, calcule
;;;; cuántos términos tiene la sucesión anterior empezando en ese
;;;; número y terminando en 1.

;;; Ejemplos:
; (contar-términos 2) ==> 2
; (contar-términos 3) ==> 8
; (contar-términos 5) ==> 6


;;; EJERCICIO 6
;;; La sucesión de Fibonacci se define por recursión como sigue:

;;; a_0 = 0
;;; a_1 = 1
;;; a_n = a_(n-2) + a_(n-1)

;;; Definir una función que, dado un número entero no negativo n,
;;; devuelva el n-ésimo término de la sucesión de Fibonacci.

;;; Ejemplos:
; (calcular-término-fib 2) ==> 1
; (calcular-término-fib 3) ==> 2
; (calcular-término-fib 10) ==> 55


;;; EJERCICIO 7
;;; Definir una función que, dada una lista de números, devuelva la
;;; media aritmética de esos números (1, en caso de ser la lista
;;; vacía).
```

```
;;; Ejemplos:
; (calcular-media-arit ()) ==> 1
; (calcular-media-arit '(1 2 3)) ==> 2
; (calcular-media-arit '(2.3 -4.85 3.75 10.0)) ==> 2.8


;;; EJERCICIO 8
;;; La progresión aritmética de primer término a_0 y diferencia d es la
;;; sucesión a_0, a_0 + d, a_0 + 2d, ...
;;; Definir una función que, dados el primer término, la diferencia y
;;; un número de términos, devuelva una lista con esos términos de la
;;; progresión aritmética correspondiente.

;;; Ejemplos:
; (construir-progresión-arit 0 1 10) ==> (0 1 2 3 4 5 6 7 8 9)
; (construir-progresión-arit 2 2 5) ==> (2 4 6 8 10)
; (construir-progresión-arit 0 -1.5 8) ==> (0 -1.5 -3.0 -4.5 -6.0 -7.5 -9.0 -10.5)


;;; EJERCICIO 9
;;; Definir una función que, dados un número y una lista de números en
;;; orden creciente, devuelva una lista en la que el número se ha
;;; insertado en el lugar que le corresponde según el orden.

;;; Ejemplos:
; (insertar-en-orden 4 '(1 2 3 5 6 7)) ==> (1 2 3 4 5 6 7)
; (insertar-en-orden 3.15 '(3.0 3.1 3.2 3.3)) ==> (3.0 3.1 3.15 3.2 3.3)


;;; EJERCICIO 10

;;; Las Torres de Hanoi es un juego matemático. Consiste en tres
;;; varillas verticales (I, C, D) y un número indeterminado de discos
;;; que determinarán la complejidad de la solución. No hay dos discos
;;; iguales, están colocados de mayor a menor en la primera varilla
;;; ascendentemente, y no se puede colocar ningún disco mayor sobre
;;; uno menor a él en ningún momento. El juego consiste en pasar todos
;;; los discos a la tercera varilla colocados de mayor a menor
;;; ascendentemente.

;;; Definir una función que, dado el número de discos, devuelva una
;;; lista de pares punteados representando la solución óptima del
;;; juego (que se puede calcular fácilmente por recursión) para ese
;;; número de discos.

;;; Ejemplos:
; (solucionar-torres-Hanoi 1) ==> ((I . D))
; (solucionar-torres-Hanoi 2) ==> ((I . C) (I . D) (C . D))
; (solucionar-torres-Hanoi 3) ==> ((I . D) (I . C) (D . C) (I . D) (C . I) (C . D) (I . D))
```