



DEPARTAMENTO DE INFORMÁTICA  
UNIVERSIDAD CARLOS III DE MADRID

# Ingeniería en Informática

## Paradigmas de Programación

Marzo 2008

### Hoja de Ejercicios 1: Problemas básicos de Lisp

#### Comentarios generales sobre los ejercicios

- Es una buena idea pensar y razonar las soluciones antes de intentar programarlas
- Para la resolución de los problemas, sólo pueden emplearse las funciones de LISP vistas en teoría
- Describir las soluciones a los ejercicios de la manera más formal posible

1. Definir un predicado que devuelva T si el menor de sus dos argumentos es par, y NIL en otro caso

---

```
(defun par-menor-p (x y)
  (evenp (if (< x y)
            x
            y)))
```

```
; ; Otra alternativa, evaluando el mínimo directamente con 'min'
(defun par-menor-1-p (x y)
  (evenp (min x y)))
```

---

2. Escribir una función cuadrado que devuelva el cuadrado de un número entregado como argumento

---

```
(defun cuadrado (n)
  (and (numberp n) (* n n)))
```

---

3. Escribir una función que transforme una temperatura en grados centígrados a grados Kelvin, y otra que realice el proceso inverso<sup>1</sup>

---

```
; Paso de grados Centígrados a Kelvin
(defun centigrados-kelvin (grados)
  (+ grados 273))
```

```
; Paso de grados Kelvin a Centígrados
(defun kelvin-centigrados (grados)
  (- grados 273))
```

---

<sup>1</sup>La temperatura en grados centígrados es la temperatura en grados Kelvin más 273

- 
4. Escribir la función `rotar` que rota los elementos de una lista hacia la derecha o hacia la izquierda

```
> (rotar 'derecha '(1 2 3 4))
> (4 1 2 3)
> (rotar 'izquierda '(1 2 3 4))
> (2 3 4 1)
```

---

```
(defun rotar (dir lista)
  (case dir
    ('derecha (append (last lista) (butlast lista)))
    ('izquierda (append (cdr lista) (list (car lista))))
    (t (format t "%pasar direccion derecha o izquierda, no ~a" dir))))
```

- 
5. Definir una función que tome un año del calendario gregoriano y devuelva si es bisiesto o no<sup>2</sup>

---

```
(defun bisiesto (anyo)
  (and (zerop (mod anyo 4))
       (or (not (zerop (mod anyo 100)))
           (zerop (mod anyo 400)))))
```

- 
6. Definir una función que tome una fecha y una hora y devuelva el número de segundos desde el 1 de enero de 1970 a las 00:00 horas

---

```
; Definición de algunas funciones útiles
(defun segundos-minuto () 60)

(defun segundos-hora ()
  (* 60 (segundos-minuto)))

(defun segundos-dia ()
  (* 24 (segundos-hora)))

(defun segundos-mes (numero-dias)
  (* numero-dias (segundos-dia)))

(defun segundos-anyo (anyo)
  (if (bisiesto anyo)
      (+ (* 4 (segundos-mes 30)) (* 7 (segundos-mes 31)) (segundos-mes 29))
      (+ (* 4 (segundos-mes 30)) (* 7 (segundos-mes 31)) (segundos-mes 28))))

; int (día: 1-31) * int (mes: 1-12) * int (año) * int (hora: 0-23)
; * int (min: 0-59) * int (seg: 0-59) -> int
(defun segundos-1970 (dia mes anyo hora min seg)
  (segundos-1970-aux dia mes anyo hora min seg 1970))

(defun segundos-1970-aux (dia mes anyo hora min seg anyo-inicial)
```

---

<sup>2</sup>Un año es bisiesto si es divisible por 4, a menos que sea divisible por 100, pero nunca por 400

```
(if (= anyo-inicial anyo)
    (segundos-fecha dia mes anyo hora min seg)
    (+
     (segundos-anyo anyo-inicial)
     (segundos-1970-aux dia mes anyo hora min seg (1+ anyoactual))))

(defun segundos-fecha (dia mes anyo hora min seg)
  (+ (segundos-meses (1- mes) anyo)
     (* (1- dia) (segundos-dia))
     (* hora (segundos-hora))
     (* min (segundos-minuto))
     seg))

(defun segundos-meses (mes anyo)
  (if (zerop mes)
      0
      (+
       (case mes
         ((1 5 7 8 10 31) (segundos-mes 31))
         (2 (if (bisiesto anyo) (segundos-mes 29) (segundos-mes 28)))
         ((3 4 6 9 11 ) (segundos-mes 30)))
       (segundos-meses (1- mes) anyo))))
```

---