**Recursividad**

**Ejercicio 1: FIBONACCI**

(DEFUN FIBONACCI(POS)  
 (IF (= POS 1)

1

(IF (= POS 2)

2

(+ (FIBONACCI(- POS 1) (FIBONACCI(- POS 2)))

)

)

)

**Ejercicio 2:** **Verificar si un elemento pertenece a una lista**

(DEFUN pertenece (x lista)

(IF (ENDP lista)

nil

(IF (= x (CAR lista))

lista

(pertenece x (CDR lista))

)

)

)

**Ejercicio 3: Sacar todos los elementos impares de una lista**

(DEFUN OBT (A)

(IF (NULL A)

NIL

(IF ( = (MOD (CAR A) 3) 0)

(CONS (CAR A) (OBT (CDR A)))

(OBT(CDR A))

)

)

)

**Ejercicio 4: Hallar el mayor elemento de una lista**

(DEFUN MAYOR(A)

(IF (NULL A)

(0)

(IF (> (CAR A) (MAYOR (CDR A)))

(CAR A)

(MAYOR (CDR A))

)

)

)

**Ejercicio 5: Hallar la sumatoria de dichos números**

**1 + ½ + ¼ + 1/8 + 1/16 +…**

(DEFUN SUM (N)  
 (IF (= N 1)  
 1

(+ (/ 1 (expt 2 (- N 1))) (SUM (- N 1))

)

)

**Ejercicio 6: SUMAR LOS ELEMENTOS DE UNA LISTA**

(DEFUN SUMA (A)

(IF (NULL A)

0

(+ (CAR A) (SUMA (CDR A)))

)

)

**Ejercicio 7: Hallar la cantidad de elementos de una lista**

(DEFUN CANT (A)

(IF (NULL A)

0

(+ 1 (CANT(CDR A)))

)

)

**Ejercicio 8: Saber si un elemento pertenece a una lista**

(DEFUN VERIF(A X)

(IF (NULL A)

nil

(IF (= (CAR A) X)

t

(VERIF (CDR A) X)

)

)

)

**Ejercicio 9: Función para saber si dos listas son iguales**

(DEFUN IGUAL (A B)  
 ( IF ( /= ( length A) (length b) nil

(IF (and (null A) (null B) t

(IF (= (CAR A) (CAR B) (IGUAL (CDR A) (CDR B))

Nil

)

)

)

)

**Ejercicio 10: Eliminar elementos que se encuentran en posiciones pares**

(DEFUN ELIMINAP (A)

(IF (ENDP A)

Nil

(CONS (CAR A) (ELIMINAP ( CDDR A)))

)

)