LABORATORIO CALIFICADO #2

- **LISTA:**

(4 1 6 (3 6 5 2) 8 (4 1) 9 7)

- **PREGUNTA 1:**

Extraer esta lista:

((Suma de hojas pares Suma de hojas impares) 4 6 8)

Solución:

```
CL-USER 1 > (setq q '(4 1 6 (3 6 5 2) 8 (4 1) 9 7))
(4 1 6 (3 6 5 2) 8 (4 1) 9 7)

CL-USER 2 > (defun sumaimparHoja(l) (if(equal l nil) 0 (if(equal (atom(car l)) t) (if(equal (mod (car l) 2) 1) (+ »
(car l) (sumaimparHoja(cdr l))) (sumaimparHoja(cdr l))) (sumaimparHoja(cdr l)))))
SUMAIMPARHOJA

CL-USER 3 > (defun sumaparHoja(l) (if(equal l nil) 0 (if(equal (atom(car l)) t) (if(equal (mod (car l) 2) 0) (+ (c»
ar l) (sumaparHoja(cdr l))) (sumaparHoja(cdr l))) (sumaparHoja(cdr l)))))
SUMAPARHOJA

CL-USER 4 > (defun sumaimparNohoja(l) (if(equal l nil) 0 (if(equal (atom(car l)) nil) (+ (sumaimparHoja(car l)) (s»
umaimparNohoja(cdr l))) (sumaimparNohoja(cdr l)))))
SUMAIMPARNOHOJA

CL-USER 5 > (defun sumaparNohoja(l) (if(equal l nil) 0 (if(equal (atom(car l)) nil) (+ (sumaparHoja(car l)) (sumap»
arNohoja(cdr l))) (sumaparNohoja(cdr l)))))
SUMAPARNOHOJA

CL-USER 6 > (defun ejercicioUno(l) (list (list (sumaimparNohoja l) (sumaparNohoja l)) (sumaimparHoja l) (sumaparHo»
ja l)))
EJERCICIOUNO

CL-USER 7 > ejercicioUno q
((9 12) 17 18)
```

- **PREGUNTA 2:**

  Extraer esta lista:

  (9 7 1)

  Solución:

```
(defun aplana (lista)
(cond ((endp lista) nil)
((atom (car lista)) (cons (car lista)(aplana (cdr lista))))
(t (append (aplana (car lista)) (aplana (cdr lista))))
)
)


(defun impar (l)
        (if (eq l nil) nil
                (if (oddp (car l)) (cons (car l) (impar (cdr l)) )
                        (impar (cdr l))
                )
        )
)

(defun aux (lista)
(if(eq l nil) nil
  (cond ((equal (car l) 9) (cons 9))
   ((equal (car l) 9) (cons 7))
   (t ((equal (car l) 9) (cons 1)))

)
)
)


(defun main (lis)
(setq q (aplana lis))
(impar q)
(aux q))
)
```

- **PREGUNTA 3:**

  Extraer esta lista:

  (Suma de hojas pares – Suma de hojas múltiplos de 3)

  Solución:

```
setq q '(4 1 6 (3 6 5 2) 8 (4 1) 9 7))

(defun sumPar(l)
  (if(equal l nil) 0
    (if(equal (atom(car l)) t)
        (if(equal (mod (car l) 2) 0)
           (+ (car l) (sumPar(cdr l)))
          (sumPar(cdr l)))
      (sumPar(cdr l))
    )
  )
)


(defun sumMultiplo(l)
  (if(equal l nil) 0 sumHojaPar
      (if(equal (atom(car l)) t)
          (if(equal (mod (car l) 3) 0)
              (+ (car l) (sumMultiplo(cdr l)))
              (sumMultiplo(cdr l)))
              (sumMultiplo(cdr l))
       )
  )
)


(defun ejercicio3(l)
  (- (sumaPar l) (sumaMultiplo l)))
```

```
CL-USER 2 >
(defun sumPar(l)
  (if(equal l nil) 0
    (if(equal (atom(car l)) t)
        (if(equal (mod (car l) 2) 0)
           (+ (car l) (sumPar(cdr l)))
          (sumPar(cdr l)))
      (sumPar(cdr l))
    )
  )
)
SUMPAR

CL-USER 3 >

(defun sumMultiplo(l)
  (if(equal l nil) 0 sumHojaPar
      (if(equal (atom(car l)) t)
          (if(equal (mod (car l) 3) 0)
              (+ (car l) (sumMultiplo(cdr l)))
              (sumMultiplo(cdr l)))
              (sumMultiplo(cdr l))
       )
  )
)
SUMMULTIPLO

CL-USER 4 >

(defun ejercicio3(l)
  (- (sumaPar l) (sumaMultiplo l)))
EJERCICIO3

CL-USER 5 > ejercicio3 q
```