Resolución TP4 – Introducción a LISP

Introducción

LISP es un lenguaje basado en listas y en el uso intensivo de paréntesis. La estructura general de una instrucción en LISP es:

```
(función argumento1 argumento2 ...)
```

El primer elemento indica qué operación se realizará y los siguientes representan los datos sobre los cuales trabajará dicha operación.

En este práctico se trabajan conceptos básicos como: Asignación de variables Manejo de listas Evaluación de símbolos Condicionales Definición de funciones Además, la consigna especifica explícitamente qué funciones deben utilizarse: **SETF, SET, EVAL, FIRST, REST, IF y DEFUN**.

Por este motivo, aunque Common Lisp posee funciones más directas como `THIRD` para obtener el tercer elemento de una lista, en este práctico no se utilizan, ya que el objetivo es practicar las funciones indicadas en la consigna y comprender la lógica de recorrido de listas mediante combinaciones de `FIRST` y `REST`.

Ejercicio 1

Consigna: Asigne a una variable una lista de 5 elementos.

```
        (setf lista '(10 20 30 40 50))
```

Explicación:

La función SETF permite asignar valores a variables. La variable llamada lista almacenará una lista de cinco elementos. La apóstrofe indica que la lista debe interpretarse literalmente y no ejecutarse como código.

Ejercicio 2

Consigna: Asignar a otra variable la suma de 1 más el tercer elemento de dicha lista.

```
        (setf resultado (+ 1 (first (rest (rest lista)))))
```

Explicación:

REST elimina el primer elemento de una lista y devuelve el resto. Aplicando REST dos veces se llega al tercer elemento. FIRST obtiene el primer elemento de la lista resultante y luego se suma 1.

Nota didáctica: Aunque Common Lisp posee funciones directas como THIRD para obtener el tercer elemento de una lista, en este práctico se trabaja únicamente con las funciones indicadas en la consigna. Por este motivo se utiliza la combinación de FIRST y REST, ya que permite comprender paso a paso la lógica de recorrido de listas.

Ejercicio 3

Consigna: Asigne W → X, X → Y, Y → Z usando únicamente SET.

```
(set 'w 'x)
(set 'x 'y)
(set 'y 'z)
```

Explicación:

SET asigna valores a símbolos. En este caso se construye una cadena de referencias: W apunta a X, X apunta a Y e Y apunta a Z. La apóstrofe indica que se trabajará con símbolos literales.

Ejercicio 4

Consigna: Usar EVAL para que en una sola instrucción devuelva Z desde W.

```
(eval (eval (eval 'w)))
```

Explicación:

La función EVAL evalúa el contenido de un símbolo. Como previamente se definió W → X → Y → Z, cada llamada a EVAL avanza un nivel. El primer EVAL devuelve X, el segundo devuelve Y y el tercero devuelve Z.

Ejercicio 5

Consigna: Asignar tres listas distintas de dos elementos cada una.

```
(setf lista1 '(1 2))
(setf lista2 '(3 4))
(setf lista3 '(5 6))
```

Explicación:

Cada variable almacena una lista distinta. Estas listas luego serán utilizadas para probar diferentes métodos de concatenación.

Ejercicio 6

Consigna: Concatenar las listas usando CONS, APPEND y LIST.

```
CONS:
(cons lista1 (cons lista2 lista3))

APPEND:
(append lista1 lista2 lista3)

LIST:
(list lista1 lista2 lista3)
```

Explicación:

CONS agrega elementos al inicio de una estructura. APPEND une varias listas en una sola lista continua. LIST crea una nueva lista cuyos elementos son las listas originales.

Ejercicio 7

Consigna: Asignar tres átomos numéricos diferentes a tres variables.

```
(setf a 5)
(setf b 8)
(setf c 3)
```

Explicación:

En este ejercicio se asignan valores numéricos simples. Los números no necesitan apóstrofe porque ya representan valores literales.

Ejercicio 8

Consigna: Usar IF para sumar 1 a la primer variable si es menor que la segunda, o sumar 2 a la segunda en caso contrario.

```
(if (< a b)
    (+ a 1)
    (+ b 2))
```

Explicación:

IF evalúa una condición. Si la condición es verdadera se ejecuta la primera opción; en caso contrario se ejecuta la segunda. Aquí se verifica si a es menor que b.

Ejercicio 9

Consigna: Usar IF con múltiples condiciones.

```
(if (and (< a b) (< b c))
    (+ a 1)
    (if (and (> b a) (> a c))
        (+ b 2)
        (+ c 3)))
```

Explicación:

AND exige que todas las condiciones sean verdaderas. Primero se evalúa si a < b y b < c. Si no se cumple, se evalúa la segunda condición. Si ninguna condición resulta verdadera, se ejecuta la última opción.

Ejercicio 10

Consigna: Crear una función que reciba dos parámetros y devuelva la suma.

```
(defun sumar (x y)
  (+ x y))
```

Explicación:

DEFUN se utiliza para definir funciones. SUMAR es el nombre de la función y x e y representan los parámetros recibidos. La expresión (+ x y) realiza la suma de ambos valores.
