Trabajo Práctico 3. Programación Funcional -Comisión C

Puntuación

Puntaje Total: 100 puntos
Aprobación: 60 puntos
Fecha de entrega: 20/11/2025 – 20:30hs

Condiciones de entrega

1. El presente trabajo práctico deberá resolverse en grupo de 3 (tres) o 4 (cuatro) integrantes, preferentemente respetando los grupos del TP1 y TP2, informar cualquier cambio de grupo.
2. Entrega: Se realizará por medio del Campus Virtual de la UTN, en la tarea correspondiente al TP 3 de la Comisión C. La extensión del archivo será .rkt. El nombre del archivo se consigue concatenando un prefijo del número del TP con los apellidos de los integrantes separados por guiones (Ej: Pérez y Abdala, el nombre será tp3-abdala-perez.rkt). Note que no hay espacios en blanco ni acentos en el nombre de archivo. En el archivo de entrega, deben constar los siguientes:
    - Primeras líneas comentadas con una línea para cada integrante, en la cual figure el nombre del alumno/a y su dirección de email.
    - Código fuente con las funciones Scheme solicitadas debidamente comentado.

Enunciado

Trabajaremos con una base de datos veterinaria donde cada animal se representa como una lista con 5 elementos:

(id nombre (clasificaciones...) (caracteristicas...) edad)

Donde:
- id: número entero único
- nombre: string con el nombre del animal
- clasificaciones: lista de strings (ej: "mamifero", "domestico")
- caracteristicas: lista de strings (ej: "vacunado", "esterilizado")
- edad: número entero (años)

BASE DE DATOS DE EJEMPLO:

```
(define bd-animales
  '((1 "Max" ("mamifero" "domestico") ("vacunado" "esterilizado") 5)
    (2 "Luna" ("mamifero" "domestico") ("vacunada") 3)
    (3 "Pipo" ("ave" "domestico") ("parlanchin") 2)
    (4 "Rex" ("mamifero" "domestico") ("guardian") 7)
    (5 "Coco" ("ave" "domestico") () 1)
    (6 "Simba" ("mamifero" "salvaje") ("carnivoro" "territorial") 8)
    (7 "Kaa" ("reptil" "salvaje") ("venenoso") 12)
    (8 "Zara" ("ave" "salvaje") ("rapaz" "nocturna") 4)))
```

EJERCICIOS A RESOLVER

---

1. OBTENER EDAD DE UN ANIMAL

Definir la función *edad* que recibe un animal (con la representación indicada antes) y retorna su edad.

Ejemplo de uso:

```
(edad '(1 "Max" ("mamifero" "domestico") ("vacunado" "esterilizado") 5))
=> 5

(edad (car bd-animales))
=> 5
```

---

2. OBTENER CARACTERÍSTICAS DE UN ANIMAL

Definir la función *características* que recibe un animal y retorna una lista con sus características.

Ejemplo de uso:

```
(caracteristicas '(1 "Max" ("mamifero" "domestico") ("vacunado" "esterilizado") 5))
=> ("vacunado" "esterilizado")

(caracteristicas (car bd-animales))
=> ("vacunado" "esterilizado")
```

---

3. BUSCAR ANIMAL POR ID

Definir la función *buscar-por-id* que recibe una base de datos y un id, y retorna el animal con ese id (o #f si no existe).

Ejemplo de uso:
(buscar-por-id bd-animales 2)
=> (2 "Luna" ("mamifero" "domestico") ("vacunada") 3)

(buscar-por-id bd-animales 10)
=> #f

---

4. LISTAR NOMBRES DE ANIMALES
Definir la función *listar-nombres* que recibe una base de datos y retorna una lista con los nombres de todos los animales de la base de datos.

Ejemplo de uso:
(listar-nombres bd-animales)
=> ("Max" "Luna" "Pipo" "Rex" "Coco" "Simba" "Kaa" "Zara")

---

5. CONTAR ANIMALES ADULTOS
Definir la función *contar-adultos* que recibe una base de datos y un número n, y cuenta cuántos animales tienen edad mayor a n.

Ejemplo de uso:
(contar-adultos bd-animales 3)
=> 5 ; Max (5), Rex (7), Simba (8), Kaa (12) y Zara (4)

(contar-adultos bd-animales 7)
=> 2 ; Simba (8) y Kaa (12)

---

6. VERIFICAR SI UN ANIMAL TIENE UNA CLASIFICACIÓN
Definir la función *tiene-clasificacion?* que recibe un animal y una clasificación, y retorna #t si el animal tiene esa clasificación, #f en caso contrario.

Ejemplo de uso:
(tiene-clasificacion? (car bd-animales) "mamifero")
=> #t

(tiene-clasificacion? (car bd-animales) "ave")
=> #f

---

7. FILTRAR ANIMALES POR CLASIFICACIÓN

Definir la función *filtrar-por-clasificacion* que recibe una base de datos y una clasificación, y retorna una lista con los animales que tienen esa clasificación.

Ejemplo de uso:

```
(filtrar-por-clasificacion bd-animales "ave")
=> ((3 "Pipo" ("ave" "domestico") ("parlanchin") 2)
   (5 "Coco" ("ave" "domestico") () 1)
   (8 "Zara" ("ave" "salvaje") ("rapaz" "nocturna") 4))

(filtrar-por-clasificacion bd-animales "salvaje")
=> ((6 "Simba" ...) (7 "Kaa" ...) (8 "Zara" ...))
```

---

8. ENCONTRAR EL ANIMAL MÁS VIEJO

Definir la función *animal-mas-viejo* que recibe una base de datos (no vacía) y retorna el animal con mayor edad.

Ejemplo de uso:

```
(animal-mas-viejo bd-animales)
=> (7 "Kaa" ("reptil" "salvaje") ("venenoso") 12)
```

---

9. CONTAR ANIMALES QUE CUMPLEN CONDICIÓN (ORDEN SUPERIOR)

Definir la función *contar-si* que recibe un predicado (función que retorna #t o #f) y una base de datos, y retorna la cantidad de animales que cumplen esa condición.

Ejemplo de uso:

```
; Contar animales mayores a 5 años
(contar-si (lambda (animal) (> (edad animal) 5)) bd-animales)
=> 3  ; Rex (7), Simba (8), Kaa (12)

; Contar animales domésticos
(contar-si (lambda (animal) (tiene-clasificacion? animal "domestico")) bd-animales)
=> 5  ; Max, Luna, Pipo, Rex, Coco

; Contar aves
(contar-si (lambda (animal) (tiene-clasificacion? animal "ave")) bd-animales)
=> 3  ; Pipo, Coco, Zara
```