INF-253 Lenguajes de Programación
Tarea 4: Scheme

Profesor: José Luis Martí Lara
Ayudante Cátedras: Sebastián Godínez San Martín
Ayudante Tareas: Gabriel Carmona Tabja - Sebastián Campos Muñoz

1. Objetivos

Conocer y aplicar correctamente los conceptos y técnicas del paradigma de programación funcional, utilizando el lenguaje **Scheme**.

2. Reglas

- Se presentarán 10 problemas en scheme. Cada uno posee una función a implementar, con su nombre y parámetros respectivos. Esto no restringe que se puedan crear funciones auxiliares. Cada problema debe ser resuelto por separado, en **archivos distintos**.
- Pueden crear funciones que no estén especificadas para utilizar en los problemas planteados, pero solo se revisara que la función pedida funcione y el problema este resuelto con la característica funcional planteada en el enunciado.
- Para implementar las funciones utilice DrRacket.
    - Descargar DrRacket

3. Problemas

1. **División inteligente de listas**
   - **Sinopsis**: `(div lista)`
   - **Característica Funcional**: Listas simples.
   - **Descripción**: Teniendo una lista, ordenada ascendentemente, se pide que se divida en dos de tal forma que los elementos de ambas listas resultantes sumen lo mismo. Si no es posible dividir la lista en dos tal que cumpla la condición debe retornar lista vacía.
   - **Ejemplos**:
     ```
     >(div '(4 5 6 7 8)
     ((4 5 6) (7 8))
     >(div '(1 2)
     ()
     ```
2. **Decimal a binario**
   - **Sinopsis**: `(decbin número)`
   - **Característica Funcional**: Cálculo de expresiones.
   - **Descripción**: Desarrollar la función *decbin*, la cual recibe un número en décimal y retorna su transformación a binario.
   - **Ejemplo**:
     ```
     >(decbin 432)
     ''110110000''
     ```
3. **La oruga lógica**
   - **Sinopsis**: `(oruga expr)`
   - **Característica Funcional**: Listas Simples y Operadores.
   - **Descripción**: Se le entregará una lista con dos elementos, una letra A, O o X que representa los siguientes operadores lógicos and, or y xor respectivamente, y una lista que contendrá 0's y 1's. Ahora deben implementar una función que retorne una lista la cual poseerá los resultados al aplicar la operación lógica a todos los elementos con su respectivo antecesor.

     Nota: La lista será mínimo de largo 2.
   - **Ejemplos**:
     ```
     >(oruga '(A '(0 1 1 0 1 1))
     '(0 1 0 0 1)
     >(oruga '(O '(1 1 1 1 0 0 1)))
     '(1 1 1 1 0 1)
     ```
4. **Distribución Hipergeometrica**
   - **Sinopsis**: `(hipercola x k N n)` y `(hipersimple x k N n)`
   - **Característica Funcional**: Recursividad simple y recursividad de cola.

- **Descripción**: Una función debe ser realizada con recursividad simple y la otra con recursión de cola. Las funciones deben entregar el resultado de la siguiente función:

$$f(x,k,N,n) = \frac{\binom{k}{x}\binom{N-k}{n-x}}{\binom{N}{n}}$$

  Donde los coeficientes binomiales se calculan de la siguiente forma:

$$\binom{a}{b} = \frac{a!}{b!(a-b)!}$$

- **Ejemplos**:

```
>(hipercola 1 2 4 3)
0.5
>(hipersimple 1 3 6 4)
0.2
```

5. **Función de multievaluación**

- **Sinopsis**: `((impar func) (par func) valor)`
- **Característica Funcional**: Expresiones Lambda.
- **Descripción**: Se debe implementar una función que realice lo siguiente: La función deberá aplicar dos funciones sobre un valor, si el valor es impar se evaluará dos veces la primera función con el valor y al resultado se le sumará lo obtenido al evaluar la segunda función una vez con el valor. Si el valor es par, se evaluará la segunda función dos veces sobre el valor y al resultado se le sumará lo obtenido al evaluar la primera función una vez con el valor.
- **Ejemplos**:

```
>(define i (impar (lambda (x) (* x x))))
>(define p (par (lambda (x) (* 2 x))))
>(i p 3)
87
>(i p 5)
635
>(i p 6)
576
```

6. **Hoja más lejana**

- **Sinopsis**: `(hoja árbol)`
- **Característica Funcional**: Recorrido de árbol binario.
- **Descripción**: Un árbol binario puede ser representado por una lista mediante:
  (valor_nodo árbol_izquierdo árbol_derecho)
  Por lo tanto, una hoja sería un nodo con dos hijos nulos:
  (valor_nodo () ())
  A partir de ello se le solicita desarrollar una función el cuál devuelva la hoja más lejana del árbol respecto a la raíz. La hoja más lejana es aquella que está en el nivel más bajo del árbol. En caso de existir más de una hoja en el nivel más lejano, se debe entregar una lista con las hojas más lejanas.

- **Ejemplos**:

```
>(hoja '(1 ( 2 (4 () ()) (5 () ())) (3 (6 () (7 () ())) ()))
7
```

7. **Cuadrado mágico**

   - **Sinopsis**: `(magia matriz)`
   - **Característica Funcional**: Recorrido de matriz.
   - **Descripción**: El cuadrado mágico es una matriz con número enteros positivos, puestos de tal forma que la suma de los números por columnas, filas y diagonales principales sea la misma. Su función se le entregará un cuadrado mágico que le falten números representados con un -1, a lo cual su función deberá rellenar la matriz y retornarla. Nota: La matriz será de 3x3 siempre. La matriz siempre tendrá a lo menos 2 números distintos de -1.
   - **Ejemplo**:

```
>(magia '('(4 -1 2) '(-1 5 7) '(-1 1 -1))
'('(4 9 2) '(3 5 7) '(8 1 6))
```

8. **Encontrar maestro**

   - **Sinopsis**: `(maestro nodo grafo)`
   - **Característica Funcional**: Recorrido de grafos.
   - **Descripción**: Un nodo se dice maestro de otro si la única forma de llegar al nodo que no es maestro es pasando por el nodo maestro. En otras palabras, si se elimina el nodo maestro, el nodo que no es maestro queda inaccesible (ya no se puede llegar a él). La función maestro recibirá el nodo de un grafo y determinará si posee un nodo maestro o no, en caso de tenerlo entregará el nodo maestro. Si se da el caso de que un nodo ya es inaccesible (no se puede llegar a él) se considera que no tiene maestros. En casos de que no se encuentre maestro, se debe retornar #f.
   - **Ejemplo**:

```
>(define grafo '('(1 '(2 3)) '(2 '(3 4)) '(3 '(4 5)) '(4 '(2)) '(5 '(4))))
>(maestro 5 grafo)
(3)
```

9. **Conga**

   - **Sinopsis**: `(conga lista lista2)`
   - **Característica Funcional**: Listas simples.
   - **Descripción**: La función conga recibe dos listas desordenadas que cumplen las siguientes características: Si una lista tiene una cantidad de números pares N, la otra lista tiene una cantidad de números impares N; y ambas listas tienen el mismo largo. Devuelve una lista con los elementos de las dos listas unidos e intercalados de forma que no queden dos elementos pares o impares de forma consecutiva.
   - **Ejemplos**:

```
>(conga '(4 5 6 7 8) '(4 5 6 7 9)
(4 5 4 5 6 7 6 7 8 9)
>(conga '(3 3 2 5 4 9 2 1) '(4 6 8 10 2 5 5 5)
(3 4 3 6 5 8 9 10 1 2 5 2 5 2 5 4)
```

10. **Cajeros**

    - **Sinopsis**: `(caja cajeros lista)`
    - **Característica Funcional**: Listas simples.
    - **Descripción**: La función caja recibe lo siguiente: El primer valor es una cantidad de cajeros y el segundo valor es una lista la cual contiene tiempos (en segundos y cómo números enteros). Los tiempos corresponden a cuanto se demorará un cliente en utilizar un cajero. La función debe retornar una lista la cuál señala que cajero utilizó cada cliente, se considera que un cajero se utiliza apenas se desocupa. Se recomienda pensar las listas simples cómo colas, de todas formas se puede utilizar cualquier método para resolverse.
    - **Ejemplos**:

      ```
      >(caja 3 '(406 424 87 888 871 915 516 81 275 578))
      (1 2 3 3 1 2 3 1 2 1)
      ```

4. Sobre Entrega

- Cada función que **NO** este definida en el enunciado del problema debe llevar una descripción según lo establecido por el siguiente ejemplo:

  ```
  ;;(Nombre_función parámetros)
  ;;Breve descripción bien explicada.
  ;;Que entrega
  ```

- Se debe trabajar en grupos de dos personas, cualquier cambio de grupo respecto a tareas anteriores debe ser avisado con brevedad.
- Cuidado con el orden y la identación de su tarea, puede llevar descuentos.
- **La entrega debe realizarse en tar.gz y debe llevar el nombre: Tarea4LP_RolIntegrante-1_RolIntegrante-2.tar.gz**
- **El archivo README.txt debe contener nombre y rol de los integrantes del grupo e instrucciones para la utilización de su programa en caso de ser necesarias.**
- **El no cumplir con las reglas de entrega conllevará un máximo de -30 puntos en su tarea.**
- La entrega será vía moodle y el plazo máximo de entrega es hasta el **Día X de MES a las 23:55 hrs.**.
- **Serán -10 puntos por cada hora de atraso.**
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.

5. Calificación

- Código no ordenado (-20 puntos)
- Código no comentado (-5 puntos)
- Problema resuelto (10 puntos cada uno)
- Reglas de entrega (-30 puntos)