

INF-253 Lenguajes de Programación

Tarea 4: Scheme

Profesor: José Luis Martí Lara, Roberto Diaz Urrea
Ayudantes Cátedra: Gabriela Acuña Benito, Hugo Sepúlveda Arriaza,
Lucio Fondón Rebolledo
Ayudantes Tareas: Gabriel Carmona Tabja, Domingo Benoit Cea,
Héctor Larrañaga Jorquera, Ignacio Ulloa Huenul,
Joaquín Gatica Hernández, Javier Pérez Riveros,
José Runín Basáez, Rafael Aros Soto
Rodrigo Pérez Jamett

1. Objetivos

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

2. Atrapado en un sueño

El informático anónimo muy feliz con su juego de rol, decide descansar e irse a dormir. Lamentablemente en su sueño sucede algo inesperado, Emehcs, dios de la programación funcional, se toma poder de su sueño y le dice “Si quieres despertar y sobrevivir deberás pasar mis 5 retos del poder”. Temeroso, el informático anónimo comienza su aventura para resolver los siguientes 5 grandes desafíos que le vienen.

3. Problemas

1. Creación obsesiva de un mazo de cartas

- **Sinopsis:** (mazo cartas divisor)
- **Característica Funcional:** Listas simples y operadores
- **Descripción:** se le entrega una lista de números (**cartas**) y un número (**divisor**), y procesar dicha lista chequeando elemento por elemento, si es divisible por el número **divisor**; en el caso que lo sea debe añadirlo a una nueva lista. Finalmente, debe retornar cuales fueron los números añadidos a esa nueva lista.
- **Ejemplo:**
 >(mazo '(1 2 3 4 5) 3)
 (3)
 >(mazo '(1 2 3 4 5) 2)
 (2 4)

2. Transformin transformon

- **Sinopsis:** (transformacion funcion1 funcion2 numeros)
- **Característica Funcional:** Funciones lambda.
- **Descripción:** se le entrega dos funciones lambda y una lista de números, y por cada número en la lista se debe hacer lo siguiente:
 - Aplicar la función1 y luego la función2, dando como resultado un número r_1
 - Aplicar la función2 y luego la función1, dando como resultado un número r_2

Luego se debe comparar r_1 con r_2 y el número más grande será almacenado en la misma posición donde está el número original.

- **Ejemplo:**

```
>(transformacion (lambda (x) (+ 2 x)) (lambda (x) (/ x 2)) '(2 3 4))  
(3 3.5 4)
```

3. Riemann zeta salvaje a aparecido

- **Sinopsis:** (zeta_simple i s) y (zetaCola i s)
- **Característica Funcional:** Recursión simple y recursión de cola
- **Descripción:** la función Riemann Zeta es una función que juega un rol importante en la Teoría analítica de números y corresponde a la siguiente fórmula:

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

Obviamente el programar una suma de 1 hasta infinito es algo inviable, por lo tanto a cada función se le entregará un i y s , donde i es el elemento de arriba de la sumatoria. Implementando así al final, la siguiente fórmula:

$$\zeta(s) = \sum_{n=1}^i \frac{1}{n^s}$$

Se debe implementar dos funciones que realicen la operación de esa fórmula, donde (zeta_simple i s) debe realizar recursión simple y (zetaCola i s) debe realizar recursión de cola.

- **Ejemplo:**

```
>(zeta_simple 3 2)  
1.3611111111111112  
>(zetaCola 3 2)  
1.3611111111111112
```

4. Árbol de la vida

- **Sinopsis:** (vida h arbol)
- **Característica Funcional:** manejo de listas
- **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 la cuál recibirá un número n y un árbol `arbol`, y retorne quienes son los ancestros de ese número. Se asegura que el número siempre existirá en el árbol.

Destacar que se trata de un árbol binario, no de búsqueda binaria.

- **Ejemplos:**

```
>(vida 4 '(5 (3 (2 () ())) (4 () ())) (8 (6 () ())) ()))
(5 3)
```

5. Temas de contagio

- **Sinopsis:** (contagio grafo n d)

- **Característica Funcional:** recorrido en listas

- **Descripción:** considerar que una representación posible de un grafo es mediante una lista de dos elementos, un número que corresponde al nodo y una lista que contiene los vecinos de ese.

El contagio en un grafo es lo siguiente: se parte con el nodo inicial y en un día, este contagia a sus vecinos. Luego, todos los contagiados en ese día contagian en el siguiente día a sus vecinos, y así sucesivamente.

Su objetivo es dado un grafo, un nodo inicial y un número de días d determinar quienes terminarán contagiados después de d días partiendo el contagio desde el nodo inicial n .

- **Ejemplos:**

```
>(contagio '((2 (1 3 4)) (1 (2)) (3 (2)) (4 (2))) 2 1)
(2 1 3 4)
```

4. Datos de Vital Importancia

- Cada problema debe ser resuelto por separado, en **archivos distintos**.
- Se debe programar siguiendo el paradigma de la programación funcional, no realizar códigos que siguen el paradigma imperativo. Por ejemplo, se prohíbe el uso de `for-each`.
- Todo código debe contener al principio `#lang scheme`
- Todos los archivos deben ser de la extensión `.rkt`
- Pueden crear funciones que no estén especificadas para utilizar en los problemas planteados, pero solo se revisará que la función pedida funcione y el problema este resuelto con la característica funcional planteada en el enunciado.
- Para implementar las funciones utilice DrRacket.
 - <http://racket-lang.org/download/>
- Cuidado con el orden y la indentación de su tarea, llevará descuento de lo más 20 puntos.

5. 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
```
- La entrega debe realizarse en tar.gz y debe llevar el nombre:
Tarea4LP_RolIntegrante-1.tar.gz
- El archivo README.txt debe contener nombre y rol del alumno 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 26 de Noviembre 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.

6. Calificación

- Código no ordenado (-20 puntos)
- Falta de comentarios (-5 puntos c/u)
- P1 (14 puntos)
- P2 (18 puntos)
- P3 (18 puntos)
- P4 (25 puntos)
- P5 (25 puntos)
- Reglas de entrega (-30 puntos)