INF-253 Lenguajes de Programación
Tarea 4: Scheme

Profesor: José Luis Martí Lara
Ayudante Cátedras: Lucio Fondón Rebolledo
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 5 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. El renacimiento Don Fondón

Luego de su aventura dentro de su cofre laberíntico, Don Fondón empezó a ver la vida con otros ojos. La vida que empezó a ver tiene muchas paréntesis, demasiadas. Entonces, como buen empresario obligó a que todos sus trabajadores cambien funcionalidades de la empresa a esta nueva forma de vida. Los trabajadores sin saber de que esta hablando Don Fondón le pide ayuda a ustedes para generar estas nuevas funcionalidades según el mundo nuevo que esta viendo.

4. Problemas

1. Te acuerdas del map de python?

- **Sinopsis**: `(mapi operador numero lista)`
- **Característica Funcional**: Listas simples y operadores
- **Descripción**: se le entrega un operador que puede tener los siguientes valores 'M' para multiplicación, 'S' para suma, 'R' para resta y 'D' para división, luego se le entrega un número y una lista con números, la idea es que aplique el operador con el número dado a todos los números de la lista.
- **Ejemplo**:

```
>(mapi 'S' 1 '(1 2 3 4 5))
(2 3 4 5 6)
>(mapi 'M' 2 '(1 2 3 4 5))
(2 4 6 8 10)
```

2. Hexadecimalinador

- **Sinopsis**: `(hex numero)`
- **Característica Funcional**: Cálculo de expresiones.
- **Descripción**: A partir de un número, debe devolver su transformación a Hexadecimal. Para más detalles de como entregar un número en hexadecimal revisar el siguiente enlace https://es.wikipedia.org/wiki/Sistema_hexadecimal.
- **Ejemplo**:

```
>(hex 160)
''A0''
```

3. Iteración de punto fijo

- **Sinopsis**: `(fpi funcion umbral i)`
- **Característica Funcional**: Funciones lambda.
- **Descripción**: El punto fijo de una función f es la solución a la ecuación:

$$f(x) = x$$

Para encontrar la solución iterativamente, se comienza con un valor inicial x_0 y este es evaluado en la funcion f, se observa si se cumple la ecuación anterior, en caso que no se cumpla el valor de x_0 es remplazado por $f(x_0)$ y se vuelve a repetir el mismo proceso hasta que x_i y $f(x_i)$ sean iguales.

La función `fpi` recibe una función que cumple con los criterios de convergencia, un valor inicial y un umbral que representa el valor absoluto de la diferencia admisible entre x_i y $f(x_i)$, es decir, $|f(x_i) - x_i|$. La función debe retornar las iteraciones que se tuvieron que realizar hasta alcanzar dicho umbral.

- **Ejemplo**:

```
>(fpi (lambda (x) (cos x)) 0.02 1)
8
```

4. **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)
```

5. **Road to champion**

- **Sinopsis**: `(rank número 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 reciba el árbol y un número y me entregue cuantos números son menor que el número dado.
  Recuerde que el árbol binario los números del árbol izquierdo son menores o iguales al número en el nodo actual y los números del árbol derecho son mayores al número en el nodo actual.
- **Ejemplos**:

```
>(rank 7 '(5 (3 (2 () ()) (4 () ())) (8 (6 () ()) ()))
5
```

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
```

- 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.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 8 de Julio 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)
- Código no comentado (-5 puntos)
- P1 (14 puntos)
- P2 (18 puntos)
- P3 (18 puntos)
- P4 (25 puntos)
- P5 (25 puntos)
- Reglas de entrega (-30 puntos)