INF-253 Lenguajes de Programación
Tarea 4: Scheme

1. Objetivos

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

2. (((((land)))))

Posterior a su experiencia creando un videojuego, Patode Hule todavía se siente insatisfecho, por lo que decide buscar nuevos horizontes. En su viaje, se encuentra con (((((land))))), una tierra que nunca antes había visto y era definitivamente distinta a las demás, convenciendosé para quedarse unos días. En su estadía se encuentra con un ex compañero de la universidad Cua Kcua K, este le pregunta si es que quiere ayudarlo como en los viejos tiempos a desarrollar unas funciones para su nueva librería. Patode apoderado por la nostalgia decide ayudarlo, pero como es de costumbre les pide ayuda a ustedes, ya que **Scheme** es un nuevo lenguaje para él.

3. Funciones

1. **La suma es un concepto puro**
   - **Sinopsis**: `(suma_especial l a)`
   - **Característica Funcional**: Funciones puras
   - **Descripción**: se le entrega una lista de números (`l`) y un número (`a`), debe retornar la suma de los elementos de `l` dividido por `a`. Debe truncar el valor resultante.
   - **Ejemplo**:
     ```
     >(suma_especial '(1 2 3 4) 4)
     2
     ```
2. **Juntemos dos conjuntos**
   - **Sinopsis**: `(merge_simple l1 l2 ops)` `(merge_cola l1 l2 ops)`
   - **Característica Funcional**: Listas simples, recursión simple y recursión cola
   - **Descripción**: se le entrega dos listas de números (`l1` y `l2`) y una lista de caracteres (`ops`), debe generar una nueva lista donde la posición i corresponde al resultado de aplicar la operación i al elemento de la posición i de `l1` con el elemento de la posición i de `l2`. Las operaciones posibles son: 'S' (suma), 'R' (resta) y 'M' (multiplicación).

     Se asegura que `l1` y `l2` tienen el mismo tamaño.

     Esta funcionalidad se debe implementar en dos funciones, dónde `(merge_simple l1 l2 ops)` debe realizar recursión de simple y `(merge_cola l1 l2 ops)` debe realizar recursión de cola.

- **Ejemplo**:

```
>(merge_simple '(1 2 3) '(4 5 6) '(#\S #\M #\R))
(5 10 -3)
>(merge_cola '(1 2 3) '(4 5 6) '(#\S #\M #\R))
(5 10 -3)
```

3. **Separemos dos conjuntos**

   - **Sinopsis**: `(demerge_simple l f) (demerge_cola l f)`
   - **Característica Funcional**: Funciones lambda, recursión simple y recursión cola
   - **Descripción**: se le entrega una lista de números (`l`) y una función lambda (`f`), por cada número en la lista se debe hacer lo siguiente:
     - aplicar `f`, retornando así un número `a`
     - al número actual restarle el número `a`, dando como resultado un número `b`

     Debe generar una lista que tenga dos listas adentro, donde cada una de la lista contenga los números `a` de cada número y la otra lista contenga los números `b` de cada número.

     Esta funcionalidad se debe implementar en dos funciones, dónde `(demerge_simple l f)` debe realizar recursión de simple y `(demerge_cola l f)` debe realizar recursión de cola.
   - **Ejemplo**:

```
>(demerge_simple '(1 2 3 4 5 6) (lambda (x) (quotient x 2)))
((0 1 1 2 2 3) (1 1 2 2 3 3))
>(demerge_simple '(1 2 3 4 5 6) (lambda (x) (modulo x 2)))
((1 0 1 0 1 0) (0 2 2 4 4 6))
>(demerge_cola '(1 2 3 4 5 6) (lambda (x) (quotient x 2)))
((0 1 1 2 2 3) (1 1 2 2 3 3))
>(demerge_cola '(1 2 3 4 5 6) (lambda (x) (modulo x 2)))
((1 0 1 0 1 0) (0 2 2 4 4 6))
```

4. **Conjunto si mismo superior**

   - **Sinopsis**: `(superior l ops f num)`
   - **Característica Funcional**: Listas simples, inmutabilidad y funciones lambda
   - **Descripción**: se le entrega una lista de números (`l`), una lista de caracteres (`ops`), una función lambda (`f`) y un número (`num`). A `l` debe aplicarle la función `merge` (función 2, puede usar cualquiera de las dos) entregando los siguiente parámetros: `l`, `l` y `ops`, dando como resultado una lista `merge_res`. Luego, a (`merge_res`) debe aplicarle la función `demerge` (función 3, puede usar cualquiera de las dos), dando como resultado una lista que contiene dos listas de números llamada `demerge_res`.

     Finalmente, debe utilizar la función `suma_especial` (función 1) de la siguiente manera a ambos resultados:

     - aplicar `suma_especial` directamente a `l` y multiplicarlo por 2, dando como resultado `r1`
     - aplicar `suma_especial` a cada una de la listas en `demerge_res` y sumar sus valores, dando como resultado `r2`

     Para así comparar `r1` con `r2`:

- si `r1` es estrictamente mayor, entonces `l` es un conjunto "si mismo superior" por lo que la función debe retornar 1.
- si `r1` no es estrictamente mayor, entonces `l` no es un conjunto "si mismo superior" por lo que la función debe retornar 0.

- **Ejemplo**:

```
>(superior '(1 2 3) '(#\S #\M #\R) (lambda (x) (- x 2)) 2)
1
>(superior '(1 2 3) '(#\S #\S #\M) (lambda (x) (modulo x 2)) 2)
0
```

5. **Matriz de conjuntos si mismo superiores**

   - **Sinopsis**: `(all_superior matriz_ls matriz_ops matriz_f matriz_nums c f)`
   - **Característica Funcional**: Manejo de listas
   - **Descripción**: una matriz se puede definir como una lista de lista de elementos de la siguiente forma:

     `((e1 e2 e3 e4) (e5 e6 e7 e8))`

     En este ejemplo se puede decir que `e1` esta en la posición (0, 0), `e2` esta en la posición (0, 1) y así con el resto de elementos.

     A partir de ello se le solicita desarrollar una función la cual recibirá una matriz de listas de números (`matriz_ls`), una matriz de listas de caracteres (`matriz_ops`), una matriz de listas de funciones lambda (`matriz_f`), una matriz de números (`matriz_nums`), un número correspondiente a la cantidad de columnas en cada matriz (`c`) y un número correspondiente a la cantidad de filas en cada matriz (`f`).

     Considerando la posición (i, j), se le debe aplicar la función `superior` (función 4) a la lista de números en esa posición con su respectiva lista de caracteres, función lambda y número.

     Realice esto por cada posición posible y así dando como resultado a un matriz de 0 y 1 correspondiente al resultado que entregue la función `superior`.

   - **Ejemplos**:

```
>(all_superior
'(((1 2 3) (1 1 1)) ((2 2 2) (3 4 5)))
'(((#\S #\M #\R) (#\S #\S #\S)) ((#\R #\R #\R) (#\S #\S #\M)))
(list (list (lambda (x) (- x 2)) (lambda (x) (modulo x 2))) (list (lambda
(x) (quotient x 2)) (lambda (x) (modulo x 2))))
'((2 3) (2 2))
2 2
)
((1 0) (1 0))
```

4. Datos de Vital Importancia

- Todo debe ser entregado en el mismo archivo donde cada función debe estar implementada en el orden que se describe en el anunciado, en el aula sección Tareas se puede encontrar un template para utilizar.
- 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 **Viernes 17 de Junio 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)
- Reglas de entrega (-30 puntos)
- F1 (10 pts)
  - Tiene implementada la función 1 de manera correcta lógicamente, pero entrega resultados correctos al momento de utilizarla (MAX 1 pts)
  - Tiene implementada la función 1 de manera correcta lógicamente y entrega resultados parcialmente correctos (MAX 5 pts)
  - Tiene implementada la función 1 de manera correcta lógicamente y entrega resultados correctos (MAX 10 pts)

- F2 (20 pts)
  - Tiene implementada una de las funciones 2 pedidas de manera correcta lógicamente, pero no entrega resultados correctos (MAX 2 pts)
  - Tiene implementada **UNA** de las funciones 2 pedidas de manera correcta lógicamente y entrega resultados correctos (MAX 10 pts)
  - Tiene implementada **AMBAS** de las funciones 2 pedidas de manera correcta lógicamente y entrega resultados correctos (MAX 20 pts)
- F3 (20 pts)
  - Tiene implementada al menos una de las funciones 3 pedidas de manera correcta lógicamente, pero no entrega resultados correctos (MAX 2 pts)
  - Tiene implementada **UNA** de las funciones 3 pedidas de manera correcta lógicamente y entrega resultados correctos (MAX 10 pts)
  - Tiene implementada **AMBAS** de las funciones 3 pedidas de manera correcta lógicamente y entrega resultados correctos (MAX 20 pts)
- F4 (25 pts)
  - Tiene implementada la función 4 de manera correcta lógicamente, pero entrega resultados correctos al momento de utilizarla (MAX 3 pts)
  - Tiene implementada la función 4 de manera correcta lógicamente y entrega resultados parcialmente correctos (MAX 15 pts)
  - Tiene implementada la función 4 de manera correcta lógicamente y entrega resultados correctos (MAX 25 pts)
- F5 (25 pts)
  - Tiene implementada la función 4 de manera correcta lógicamente, pero entrega resultados correctos al momento de utilizarla (MAX 3 pts)
  - Tiene implementada la función 4 de manera correcta lógicamente y entrega resultados parcialmente correctos (MAX 15 pts)
  - Tiene implementada la función 4 de manera correcta lógicamente y entrega resultados correctos (MAX 25 pts)