INF-253 Lenguajes de Programación
Tarea 4: Scheme

3 de noviembre de 2022

1. Objetivo

En esta tarea deberán implementar una serie de funciones para conocer y aplicar correctamente los conceptos y técnicas del paradigma de programación funcional, utilizando el lenguaje **Scheme**.

2. Funciones a implementar

1. **Inverso**
   - **Sinopsis**: `(inverso lista n)`
   - **Característica Funcional**: Funciones puras
   - **Descripción**: Se le entrega una lista de números (`lista`) y un número (`n`), debe retornar una lista con todos los números entre 0 y `n` (sin incluir) que no estén en `lista`.
   - **Ejemplo**:
     ```
     >(inverso '(1 3 7) 10)
     (0 2 4 5 6 8 9)
     ```
2. **Umbral**
   - **Sinopsis**: `(umbral_simple lista umbral tipo)` `(umbral_cola lista umbral tipo)`
   - **Característica Funcional**: Listas simples, recursión simple y recursión cola
   - **Descripción**: Se le entrega una lista de números (`lista`), un numero (`umbral`) y un carácter (`tipo`). Si `tipo` es 'M' debe retornar una lista con todas las posiciones de los elementos de `lista` que sean *Mayores* que `umbral`, si `tipo` es 'm' entonces debe retornar todas las posiciones de los elementos *menores* que `umbral`.

     Esta funcionalidad se debe implementar en dos funciones, dónde `(umbral_simple lista umbral tipo)` debe realizar recursión de simple y `(umbral_cola lista umbral tipo)` debe realizar recursión de cola.
   - **Ejemplo**:
     ```
     >(umbral_simple '(15 2 1 3 27 5 10) 5 #\M)
     (0 4 6)
     >(umbral_cola '(15 2 1 3 27 5 10) 5 #\m)
     (1 2 3)
     ```
3. **Modificar seleccionados**

- **Sinopsis**: `(modsel_simple lista seleccion f) (modsel_cola lista seleccion f)`
- **Característica Funcional**: Funciones lambda, recursión simple y recursión cola
- **Descripción**: Se le entrega dos listas de números (`lista` y `seleccion`) y una función lambda (`f`), por cada número en la lista, si su índice esta en `seleccion` entonces se le debe aplicar la función `f`, en caso contrario el numero se mantiene igual.

  Esta funcionalidad se debe implementar en dos funciones, dónde `(modsel_simple lista seleccion f)` debe realizar recursión de simple y `( (modsel_cola lista seleccion f))` debe realizar recursión de cola. S
- **Ejemplo**:

  ```
  >(modsel_simple '(15 2 1 3 27 5 10) '(0 4 6) (lambda (x) (modulo x 2)))
  (1 2 1 3 1 5 0)
  >(modsel_simple '(15 2 1 3 27 5 10) '(3 1 2) (lambda (x) (+ x 5)))
  (15 7 6 8 27 5 10)
  ```

4. **Estables**
   - **Sinopsis**: `(estables lista umbral fM fm)`
   - **Característica Funcional**: Listas simples, inmutabilidad y funciones lambda
   - **Descripción**: Se le entrega una lista de números (`lista`), un numero (`umbral`) y dos funciones lambda (`fM` y `fm`). Utilizando las funciones implementadas anteriormente, retornar una lista con dos números, en donde el primero es la cantidad de números mayores que el umbral que al aplicarles `fM` siguen siendo mayores que el umbral, y el segundo es la cantidad de números menores que el umbral que al aplicarles `fm` siguen siendo menores que el umbral.
   - **Ejemplo**:

     ```
     >(estables '(15 2 1 3 27 5 10) 5 (lambda (x) (/ x 2)) (lambda (x) (* x 2)))
     (2 1)
     ```

5. **Todo lo anterior, y ahora en 2D!**
   - **Sinopsis**: `(query lista pos op params)`
   - **Característica Funcional**: Manejo de listas
   - **Descripción**: Se le entrega una lista de listas de enteros (`lista`), la posición de una lista de enteros dentro de esta lista de listas (`pos`), un numero entre 1 y 3 que indica una operación a realizar sobre la lista (`op`) y una lista con los parámetros necesarios para esa operación (`params`).

     Para los valores de pos:
     - **1**: Debe aplicar `Umbral` sobre la lista en la posición `pos`. La variable `params` contendrá dos elementos: el valor del umbral y el tipo de umbral.
     - **2**: Debe aplicar `modsel` sobre la lista en la posición `pos`. La variable `params` contendrá dos elementos: una lista con la selección y una función a aplicar a la selección.
     - **3**: Debe aplicar `estables` sobre la lista en la posición `pos`. La variable `params` contendrá tres elementos: el valor del umbral, una función fM y una función fm.

     La función debe retornar el resultado de la operación (no la lista de listas modificada).

- **Ejemplos**:

```
>(query '((0 1 2 3 4) (4 3 2 1 0) (15 2 1 3 27 5 10)) 1 1 '(1 #\M))
(0 1 2)
>(query '((0 1 2 3 4) (4 3 2 1 0) (15 2 1 3 27 5 10)) 0 2 '((0 4) (lambda
(x) (+ x 100))))
(100 1 2 3 104)
>(query '((0 1 2 3 4) (4 3 2 1 0) (15 2 1 3 27 5 10)) 2 3 '(5 (lambda (x)
(/ x 2)) (lambda (x) (* x 2))))
(2 1)
```

3. Sobre la Entrega

- Se deberá entregar un único archivo con todas las funciones implementadas en el orden descrito en el enunciado. En aula se encuentra disponible una plantilla de como debe ser este archivo.
- 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`.
- Para implementar las funciones utilice DrRacket.
- Todo código debe contener al principio `#lang scheme`
- Se debe entregar un archivo con 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.
  - http://racket-lang.org/download/
- Cuidado con el orden y la indentación de su tarea, llevará descuento de lo más 20 puntos.
- Las funciones implementadas y que no esten en el enunciado deben ser comentadas de la siguiente forma. **SE HARÁN DESCUENTOS POR FUNCIÓN NO COMENTADA**

```
;; Descripcion de la funcion
;;
;; a: Descripcion del parametro a
;; b: Descripcion del parametro a
```

- Se debe trabajar de forma individual obligatoriamente.
- **La entrega debe entregarse en .tar.gz y debe llevar el nombre: Tarea4LP_RolAlumno.tar.gz**
- **El archivo README.txt debe contener nombre y rol del alumno e instrucciones detalladas para la correcta utilización de su programa. De no incluir README se realizara un descuento.**
- La entrega será vía aula y el plazo máximo de entrega es hasta **18 de Noviembre**.
- **Por cada hora de atraso se descontaran 20 pts.**

- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.
- **Solo se contestaran dudas realizadas en AULA y que se realicen al menos 48 horas antes de la fecha de entrega original.**

4. Calificación

4.1. Entrega

- `inverso` (**10 pts**)
- `umbral` (**20 pts**): 10 pts la función recursiva simple y 10 pts la función recursiva de cola.
- `modsel` (**20 pts**): 10 pts la función recursiva simple y 10 pts la función recursiva de cola.
- `estables` (**25 pts**)
- `query` (**25 pts**)

4.2. Descuentos

- Falta de comentarios (-5 pts c/u Max 20 pts)
- Falta de README (-20 pts)
- Falta de alguna información obligatoria en el README (-5 pts c/u)
- Falta de orden (entre -5 y -20 pts dependiendo de que tan desordenado)
- Entrega tardía (-20 pts por cada hora de atraso)
- Mal nombre en algún archivo entregado (-5 pts c/u)