INF-253 Lenguajes de Programación
Tarea 4: Scheme

Profesor: Roberto Diaz
Ayudante Cátedra: Anastasiia Fedovora
Ayudante Tareas: Héctor Larrañaga - Sebastián Campos - Axel Reyes

20 de noviembre de 2020

1. Lore

Con mucho esfuerzo, paciencia y sueño lograste derrotar a las temidas bestias del mundo POO. Lamentablemente, en la batalla perdiste tus indentaciones y tus `;`, lo cual te deja desorientado e incapaz de seguir recorriendo largas distancias sin perderte en el camino en aquel lugar tan hostil. Estás al borde de la desesperación cuando a lo lejos reconoces a un viajero con una gran mochila que, exhausto, camina lentamente en este lugar desértico. Con esperanza en la mirada, te acercas a su posición en búsqueda de ayuda. Tímidamente, le intentas preguntar su nombre, pero él te mira desde la oscuridad de su capucha en silencio y con un gesto lento por el cansancio te apunta a sus pies. Sorprendido, bajas la vista y te das cuenta con mucha tristeza que la persona frente a ti tiene la marca '`( )`' propia de los condenados funcionales. Los condenados funcionales son aquellas personas que perdieron el Desafío Funcional. Este es un desafío engañoso pues te promete una gran recompensa, pero si no logras superarlo estarás condenado a una vida de '`( )`' para siempre. Como todo condenado funcional, el desconocido te propone probar tu suerte en el desafío. En caso de superarlo, promete cumplirte cualquier deseo que tengas, pero si fallas tu destino estará sellado.

Sin ninguna otra opción y con el deseo de salir de este planeta, *aceptas el reto*.

2. Instrucciones

- 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** y con nombres `p1.rkt` para el problema 1, `p2.rkt` para el problema 2, etc.
- 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 esté resuelto con la característica funcional planteada en el enunciado.
- Para implementar las funciones utilice DrRacket.
  - Descargar DrRacket desde http://racket-lang.org/download/

3. Problemas

A continuación se presentan los cinco problemas a resolver. Asuma que los inputs serán del tipo de dato solicitado.

1. **Ser flojo tiene sus beneficios**
   - **Sinopsis**: `(lazypascal n)`
   - **Característica Funcional**: Evaluación Perezosa.
   - **Descripción**: Se debe implementar la función lazypascal la cual recibe un entero n y retorna el n-ésimo piso del triangulo de Pascal. El cálculo debe utilizar Evaluación Perezosa.
   - **Ejemplos**:

     ```
     >(lazypascal 5)
     (1 5 10 10 5 1)
     >(lazypascal 8)
     (1 8 28 56 70 56 28 8 1)
     ```

2. **El mundo se consume en dinero**
   - **Sinopsis**: `(monedas_iter l1)` `(monedas_rec l1)`
   - **Característica Funcional**: Recursión vs Iteración.
   - **Descripción**: Se deben implementar dos funciones. Ambas reciben una lista que contiene monedas de diversos valores representadas cada una por un número entero y se debe retornar la suma máxima de monedas posible, bajo la condición de que no se pueden tomar dos monedas adyacentes. Una de las funciones debe implementarse con recursión, mientras que la otra usando iteración.
   - **Ejemplos**:

     ```
     >(monedas_iter '(5 22 26 15 4 3 11))
     48
     >(monedas_rec '(-3 -49))
     -3
     ```

3. **Criss Cross**
   - **Sinopsis**: `(crossrd l1 l2)`
   - **Característica Funcional**: Manejo de listas.
   - **Descripción**: La función debe recibir dos listas del mismo largo y generar dos cortes escogidas al azar, pero iguales entre las listas. Luego, debe armar dos nuevas listas intercambiando la parte central de los cortes de ambas listas. Notar que una lista de largo n tiene $n + 1$ puntos de corte.
   - **Ejemplos**:

     En este ejemplo se produce un corte antes del 4 y del 6 de la primera lista, y en las posiciones análogas en la segunda lista.

     ```
     >(crossrd '(1 2 3 4 5 6 7 8) '(9 10 11 12 13 14 15 16))
     (1 2 3 12 13 6 7 8) (9 10 11 3 4 14 15 16)
     ```

     En este ejemplo se produce un corte antes del 2 y después del 3 de la primera lista, y en las posiciones análogas en la segunda lista.

     ```
     >(crossrd '(1 2 3) '(8 10 5))
     (1 10 5) (8 2 3)
     ```

4. **Scheme and the blind forest.**

   - **Sinopsis**: `(preorden arbol funcion)`
   - **Característica Funcional**: Forma funcional.
   - **Descripción**: Se debe implementar una función que reciba un árbol y una función. Debe aplicar la función sobre un recorrido preorden del árbol.

     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 () ())`
   - **Ejemplos**:

     ```
     >(preorden '(7 (5 (1 () ()) (2 () ())) (6 (3 () ()) (4 () ()))) (lambda
     x x))
     (7 5 1 2 6 3 4)
     >(preorden '(7 (5 (1 () ()) (2 () ())) (6 (3 () ()) (4 () ()))) (lambda
     x (map sqrt x)))
     (2.6457513110645907 2.23606797749979 1 1.4142135623730951 2.449489742783178
     1.7320508075688772 2)
     ```

5. **McLaurin Supercool**

   - **Sinopsis**: `(seno x)`
   - **Característica Funcional**: Recursión de cola.
   - **Descripción**: Se debe implementar una función que aproxime el cálculo de la función seno. Para esto se utilizará el teorema de Maclaurin, obteniendo la siguiente serie de potencias para el seno:

     $$sen(x) = \sum_{n=0}^{\infty} (-1)^n \cdot \frac{x^{2n+1}}{(2n+1)!}$$

     La implementación debe utilizar recursión de cola.
   - **Ejemplos**:

     ```
     >(seno 0)
     0
     >(seno 0.56)
     0.53118619792
     >(seno 90)
     0.8939966636
     ```

4. Sobre Entrega

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

  ```
  ;;(Nombre_función parámetros)
  ;;Breve descripción de su funcionamiento.
  ;;Retorno de la función.
  ```

- Se debe trabajar de forma individual.
- Cuidado con el orden y la indentación de su tarea, puede llevar descuentos.
- **La entrega debe realizarse en .zip y debe llevar el nombre: Tarea4LP_RolIntegrante-1.zip**
- **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 o formato conllevará un descuento de máximo 30 puntos en su tarea.**
- La entrega será vía aula y el plazo máximo de entrega es hasta el **día 12 de diciembre a las 23:55 hrs.**.
- Sólo se contestarán dudas **hasta 3 días** antes de la fecha de entrega original.
- **Habrá un descuento de 20 puntos por cada día de atraso.**
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.

5. Calificación

- Problema 1: 30 pts
- Problema 2: 25 pts
- Problema 3: 15 pts
- Problema 4: 15 pts
- Problema 5: 15 pts
- Código no ordenado (-20 puntos)
- Código no comentado (-5 puntos)
- Reglas de entrega (-30 puntos MAX)