PEP 1 Paradigmas de Programación (2021-1)

NOMBRE:_____ RUT: _____ Profesor: _____

Instrucciones

- Tiempo estimado de respuesta: 90 minutos
- Tiempo disponible para entregar respuestas desde que tuvo acceso al enunciado: 3 horas hasta las 23:59 del 13 de mayo de 2021.
- Lugar de entrega: Campus Virtual.
- Escriba sus respuestas en hojas separadas
- Puede escribir sus respuestas directamente en un documento electrónico o bien puede hacerlo en papel. Escoja el que más le acomode.
- Recomendación: Para asegurar acceso continuo a este enunciado en caso de cortes de suministro eléctrico o en el acceso a Internet, procure descargar este enunciado y si puede, imprimalo.
- Identifique cada hoja de respuestas con su nombre y rut, lo que sirve de complemento a la declaración que ha aceptado en Campus Virtual.
- Identifique cada hoja de respuesta con el profesor de su sección.
- Al terminar la evaluación,
    - Si desarrolló sus respuestas con lápiz y papel, saqué una fotografía o escanee cada una de sus hojas de respuestas. Asegúrese que las imágenes de sus capturas son legibles (abra el archivo y verifique). Luego, agrupe todas las imágenes en un PDF (en Windows basta con seleccionar las imágenes desde el explorador de archivos -> clic derecho -> Imprimir -> Escoger Impresora PDF de Microsoft u otra que disponga) o bien en un archivo comprimido *.zip* o *.rar*.
    - Si desarrolló sus respuestas directamente en formato digital (ej: documento Word, Google Docs, Latex, etc.) guardar el documento como PDF. Se recomienda este formato para evitar problemas en la visualización de estilos.
- Finalmente, subir el archivo al espacio creado para tales efectos en Campus Virtual. Una vez subido el archivo, descárguelo y asegúrese que el archivo se puede abrir (para evitar situaciones de archivos dañados).
- Excepción: En la eventualidad que no pueda subir su trabajo a Campus Virtual en el plazo disponible, enviarlo inmediatamente vía correo a su profesor.

Sobre la evaluación. Todo ítem se evalúa en una escala de 3 puntos: 0 (sin respuesta o respuesta no aborda el problema), 0.5, 1 (respuesta completa sin errores), según grado de cumplimiento.

NOTA 1: Procure hacer un adecuado uso del paradigma funcional (principalmente declarativo en la medida de lo posible) de manera de no redundar en esfuerzos de implementación. Si lo estima conveniente, puede implementar funciones propias del TDA y otras funciones complementarias que pueden ser usadas en ambas preguntas. La implementación innecesaria y/o redundante de funciones que desaprovechen el paradigma funcional tendrán una penalización de hasta **2 puntos** por cada ítem afectado.

NOTA 2: Recuerde documentar todas sus funciones indicando dominio, recorrido, descripción de su funcionalidad, tipo de recursión (si aplica) y ejemplo de uso (una sola línea, es un ejemplo de cómo se evalúa ej: (función 3 4 5)). El no hacerlo puede significar un descuento de hasta **1 punto** por cada ítem afectado.

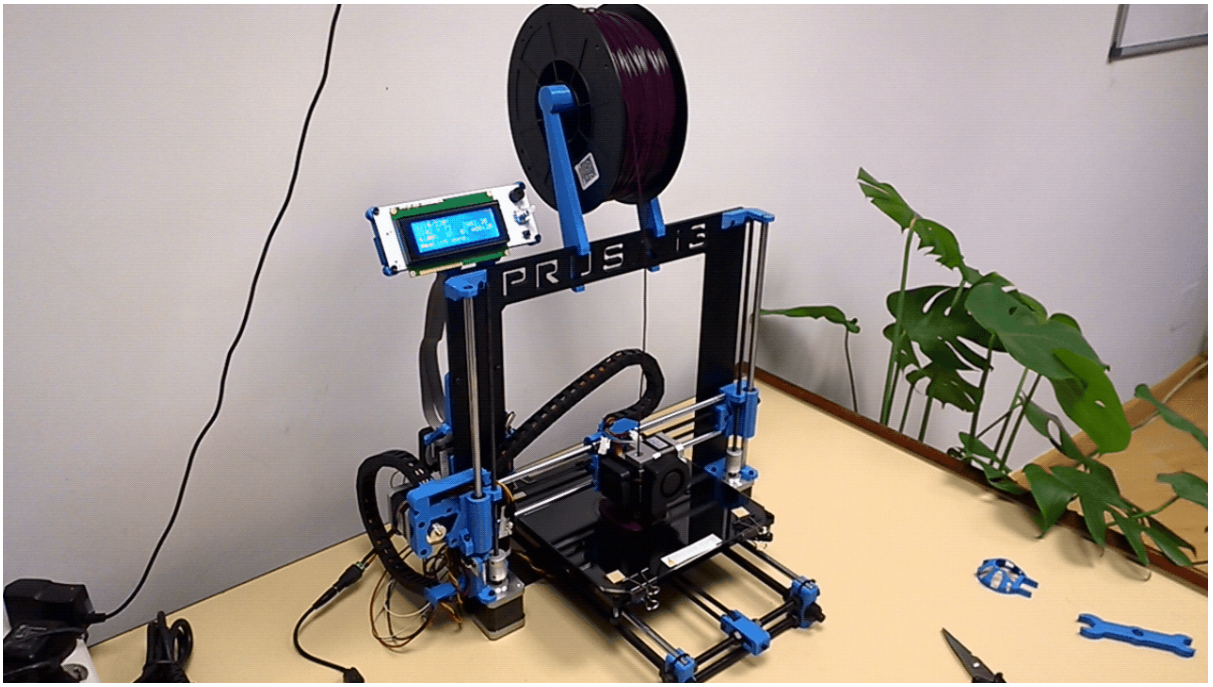
NOTA 3: Para cada pregunta, puede usar las siguientes funciones de Scheme: car, cdr, if, cond, define, let, null?, =, eqv?, eq? equal?, +, *, /, -, list?, null, let, lazy, lambda, force, not, and, or, eq?, cons, list, round (para redondear), length. Cualquier otra función no indicada en esta lista deberá ser implementada.

Pregunta 1 (30 pts)

Las impresoras 3D de tipo FDM se basan en el uso de un cabezal de impresión que deposita un filamento de plástico derretido sobre una superficie capa a capa. El cabezal tiene la capacidad de moverse en el eje X y en el eje Z, mientras que la base de impresión puede moverse en el eje Y.

Para poder imprimir un modelo 3D este debe estar en un formato que acepte la impresora, este formato se basa en instrucciones que le dicen a la impresora las acciones que debe realizar, como por ejemplo:

- Moverse en X e Y cierta cantidad de milímetros
- Moverse en Z cierta cantidad de milímetros
- Extruir o retraer filamento (Acciona el motor que empuja el filamento de plástico para que salga plástico por la boquilla) cierta cantidad de milímetros.



Por ejemplo, para dibujar una línea diagonal de 5mm de largo y 0.6mm de alto usando una altura de capa de 0.3mm, a la impresora se deben enviar los siguientes comandos (Es cómo un lenguaje *assembler* - imperativo para las impresoras 3D):

```
EXTRUIR 0.4 # Empuja 0.4mm de filamento por el cabezal de impresión
MOVER 3 4 # Se mueve en diagonal 3mm en X y 4mm en Y (diagonal de 5mm)
SUBIR 0.3 # Sube 0.3mm en Z
EXTRUIR 0.4 # Empuja más filamento
MOVER -3 -4 # Para devolverse en la misma diagonal
EXTRUIR 0 # Deja de presionar la salida del filamento
SUBIR 10 # Par dejar el cabezal 1cm sobre la figura ya terminada
```

Usted está encargado de un módulo del software utilizado para convertir un diseño 3D al formato aceptado por las impresoras, en este módulo se deben obtener estadísticas acerca de la impresión realizada.

a) **(5 pts)** Diseñe el/los TDAs necesario(s) para representar los comandos que serán enviados a la impresora 3D, implemente las funciones del TDA que pueda necesitar en las siguiente preguntas.

El TDA “comando” lo defino como una lista con el nombre del comando y luego una lista de argumentos. Ej: ‘(“EXTRUIR” (0.4))

Y el listado de comandos es sólo una lista del TDA descrito anteriormente.

```
(define comando->getNombre car)

(define comando->getArgs cadr)
```

b) **(5 pts)** Realice una función que permita obtener el peso total de la impresión generada. Para su función asuma que además tiene como entrada el grosor (o diámetro) en centímetros del filamento y la densidad de este (en g/cm^3).

Esta pregunta se simplifica entendiendo como que se debe realizar primero un filtro de sólo los comandos EXTRUIR, luego aplicando la función “reduce” (debe ser implementada) para obtener la cantidad de milímetros de filamento utilizado y finalmente aplicando las fórmulas para para obtener el peso de impresión.

```
(define filtar (lambda (L f) (
                        (if (null? L)
                                `()
                                (if (f (car L))
                                        (cons (car L) (filtrar (cdr L) ff))
                                        (filtrar (cdr L) f )
                                )
                        )
                )
        )
)
(define reducir (lambda (L fRedux cont)
                (if (null? L)
                        cont
                        (reducir (cdr L) fRedux (fRedux (car L) cont))
                )
        )
)
(define largoImpresion (lambda (comandos)
                (reducir
                        (map
                                (compose car comando->getArgs) ; Para obtener primer argumento
                                (filtrar comandos (lambda (c)
                                                (equal? (comando->getNombre c) “EXTRUIR”)
                                        )
                                )
                        )
                        +
                        0
                )
        )
)
(define areaTransversal (lambda (r) (* PI r r) ) )
(define pesoImpresion (lambda (comandos grosor densidad)
                (* densidad (* (areaTransversal (/ grosor 2) (largoImpresion comandos) ) ) )
        )
)
```

c) **(5 pts)** Realice una función que permita obtener el tiempo total de impresión, asuma que los comandos de EXTRUIR se ejecutan al mismo tiempo que el siguiente comando de movimiento (puede ser MOVER o SUBIR), los comandos de movimiento demoran siempre 0.1 segundos en ejecutarse.

Idea similar a lo anterior, se filtra sólo por los comandos “SUBIR” o “MOVER”, luego se usa map para obtener una lista de unos, los cuales serán sumados mediante “reducir”. Finalmente se multiplica por 0.1 que es el tiempo de cada movimiento.

```
(define (tiempoTotalImpresion comandos)
        (*
                0.1
                (reducir
                        (map
                                (lambda (c) 1) ; Siempre retorna 1
                                (filtrar comandos (lambda (c)
                                                (or
                                                        (equal? (comando->getNombre c) “SUBIR”)
                                                        (equal? (comando->getNombre c) “MOVER”)
                                                )
                                        )
                                )
                        )
                        + ; Sumo puros 1, para contar los comandos “SUBIR” o “MOVER”
                        0
                )
        )
)
```

d) **(7.5 pts)** Implemente una función de orden superior que permita obtener cualquier estadística a partir del TDA con el listado de comandos. Por ejemplo si queremos contar los comandos de tipo “SUBIR” enviados a la impresora:

```
(obtenerEstadistica comandos (lambda (x cont) (if (equal? "SUBIR" (getTipoComando x)) (+ cont 1) cont)) )
```

La función solicitada es justamente lo mismo que “reducir” que fue definida anteriormente.

```
(define obtenerEstadistica reducir)
```

e) **(7.5 pts)** Implemente la misma función que en d), pero currificada.

Se puede currificar en cualquier orden sus argumentos.

```
(define obtenerEstadisticaCurry (lambda (L)
                (lambda (fRedux)
                        (lambda (cont)
                                (obtenerEstadistica L fRedux cont)
                        )
                )
        )
)
```

Hints: Entendiendo que el filamento utilizado para la impresión es circular y su grosor queda expresado a través del diámetro de éste, considere los siguientes cálculos para abordar las funciones requeridas en la pregunta.

El área transversal del filamento se puede calcular como el área de un círculo:

$$A = PI * r^2$$

Y para obtener su volumen debe multiplicar el área transversal por el largo del filamento

$$V = A * L$$

Finalmente tenemos la fórmula de densidad que relaciona la masa y el volumen:

$$\rho = \frac{m}{V}$$

Pregunta 2 (30 pts)

Dado el contexto global de la pandemia, la librería "*Pingüino*" vio mermada su venta física de libros de programación de autores independientes que no cuentan con el apoyo de grandes editoriales como O'Reilly Publications. Es por este motivo que para impulsar la publicación y ventas de libros de este tipo, la gerencia de la empresa tiene la intención de implementar un portal en donde los usuarios tengan la posibilidad de ofrecer la venta de libros técnicos de su autoría a otros usuarios. La diferencia con la competencia es que los autores piden a los usuarios un precio mínimo y sugerido a pagar por estos libros.

Este sistema contiene el **registro de autores asociado, todos los libros publicados en el sistema "*Pingüino*" y un listado de usuarios clientes registrados**. En esta línea, un **autor** consta de nombre (*string*) y un listado de libros asociados a su autoría (*lista de libros*). Mientras que cada **libro** tiene asociado un nombre de autor (*string*), el nombre de libro (*string*), el precio mínimo (*number*) y sugerido (*number*) a pagar por este libro, la cantidad de ejemplares vendidos (*number*), el número de páginas (*number*) y un listado de tags que indican las tecnologías asociadas (lista de *strings*) (por ejemplo, "TypeScript", "Go", "Rust"). Finalmente, un **usuario** tiene un username (string) y un listado de libros comprados.

Figura: A modo de referencia visual, un libro (Aprendiendo JavaScript) publicado por un autor (Carlos Azaustre) en donde se solicita un precio mínimo y uno sugerido para adquirir esta obra. Fuente: leanpub.com

a. **(2 pts)** Especificar TDAs que permitan cubrir con lo necesario para abordar las necesidades de la editorial "*Pingüino*" de acuerdo a los requerimientos de las siguientes preguntas. Luego para la implementación del TDA señalar las representaciones escogidas para cada uno y las funciones de éste (constructores, selectores, modificadores) que resulten necesarias para abordar las siguientes preguntas.

La siguiente pauta solo describe las funciones elementales para agregar elementos en cada grupo de elementos. Se deben contextualizar las funciones en el sistema que alberga libros, autores, usuarios. Respuestas que se ajusten a está categoría solo obtienen .75 en el correspondiente item. Para puntaje completo, se necesita contextualizar en el sistema.

```
;; a) Tda necesarios (total 2 ptos)

;; TDA Autor (0.5 ptos)

(define (autor nombre libros-publicados)

 (list nombre libros-publicados))

;; TDA Libro (0.5 ptos)

(define (libro nombre-autor nombre-libro precio-minimo precio-sugerido total-vendidos numero-paginas tags)

 (list nombre-autor nombre-libro precio-minimo precio-sugerido total-vendidos numero-paginas tags))

;; TDA Usuario (0.5 pto)

(define (usuario username libros-comprados)

 (list username libros-comprados))


(define (lista-autores . autores) autores)

(define (lista-libros . libros) libros)

(define (lista-usuarios . usuarios) usuarios)

;; Selectores (0.5 pto)

;; Estos selectores eran los mínimos necesarios, solo bastaba con estos

(define (get-username usuario) (car usuario))

(define (get-libros-comprados usuario) (cadr usuario))

(define (get-precio-minimo libro) (caddr libro))

(define (get-total-vendidos libro) (caddddr libro))

(define (get-numero-paginas libro) (cadddddr libro))
```

b. **(5 pts)** Implemente las funciones necesarias que permitan registrar un nuevo autor en el sistema.

```
;; Registrar nuevo autor (total 5 ptos)

(define (registro-autor nombre libros-publicados autores)

 (if (null? autores)

     (cons (autor nombre libros-publicados) null)

     (cons (car autores) (registro-autor nombre libros-publicados (cdr autores)))

     )

 )
```

```
alternativa

(define (registro-autor nombre libros-publicados autores)

        (cons (autor nombre libros-publicados) autores))
```

c. **(5 pts)** Implemente las funciones necesarias que permitan al autor registrar un nuevo libro de su autoría en el sistema.

```
;; c) Permitir al autor registrar un nuevo libro de su autoría en el sistema (5 ptos)


(define registro-libro

   (lambda   (nombre-autor   nombre-libro   precio-minimo   precio-sugerido   total-vendidos   numero-paginas   tags
libros-publicados)

   (if (null? libros-publicados)

       (cons (libro nombre-autor

                    nombre-libro

                    precio-minimo

                    precio-sugerido

                    total-vendidos

                    numero-paginas

                    tags) null)

       (cons (car libros-publicados)

             (registro-libro nombre-autor

                             nombre-libro

                             precio-minimo

                             precio-sugerido

                             total-vendidos

                             numero-paginas

                             tags

                             (cdr libros-publicados))))))
```

;alternativa - inserta por cabeza

```
(define registro-libro

   (lambda   (nombre-autor   nombre-libro   precio-minimo   precio-sugerido   total-vendidos   numero-paginas   tags
libros-publicados)

       (cons (libro nombre-autor  nombre-libro   precio-minimo precio-sugerido total-vendidos numero-paginas tags)
```

```
                libros-publicados)))
```

d. **(5 pts)** Implemente las funciones necesarias que permitan a un usuario comprar (adquirir) un libro de la lista de libros existentes en el sistema.

```
;; d) Permitir a un usuario comprar (adquirir) un libro de la lista de libros existentes en el sistema (total 5 ptos)

;; Definir función de agregar (1 pto)
(define (agregar lista elemento)
 (if (null? lista)
     (cons elemento null)
     (cons (car lista) (agregar (cdr lista) elemento)))
 )
alternativa con inserción por cabeza
(define (agregar lista elemento) (cons elemento lista))


;; Definir función que verifica si el elemento se encuentra en la lista (1 pto)
(define (contiene? lista elemento)
 (cond
   [(null? lista) #f]
   [(equal? (car lista) elemento) #t]
   [else (contiene? (cdr lista) elemento)]))

alternativa
(define (contiene? lista elemento) (not (null? (filtrado-libros libros (lambda (l) (equals? l elemento))))))

;; Función de adquirir el libro (3 ptos)
;; Sin cambiar la cantidad de libros vendidos (2 ptos)
(define (adquirir-libro usuario-comprador libro-a-adquirir sistema)
 (if (contiene? sistema libro-a-adquirir)
    (usuario (get-username usuario-comprador)
                    (agregar (get-libros-comprados usuario-comprador) libro-a-adquirir))]
   usuario-comprador
  )
)
```

```
;; Considerando cambiar la cantidad de libros vendidos (asignar puntaje completo, 3 ptos)

(define (adquirir-libro usuario-comprador libro-a-adquirir sistema)

 (cond

   [(contiene? sistema libro-a-adquirir)

    (usuario (get-username usuario-comprador)

             (agregar (get-libros-comprados usuario-comprador)

                      (libro (+ 1 (get-total-vendidos libro-a-adquirir) total-vendidos)

                             ; Constructor de libro pero con el totalvendidos + 1

                             )

                      ))]

   [usuario-comprador]))
```

e. **(13 pts)** Implemente las funciones necesarias que permitan filtrar libros en el sistema de acuerdo a:
   i. precio mínimo **(3 pts)**
   ii. precio máximo **(3 pts)**
   iii. número de páginas **(3 pts)**
   iv. un valor entre el precio mínimo y el precio máximo **(4 pts)**

```
; Esta pregunta se puede abordar creando una función de filtro

; En esta pregunta se le da la libertad al estudiante de como querer implementar los criterios y los filtros necesarios. No hay una instrucción específica para por ejemplo implementar un criterio basado en precio mínimo, queda a disposición del estudiante qué hacer con ese precio mínimo, máximo, numero de paginas, entre otros

(define (filtro-libros libros criterio)

 (if (null? libros)

     null

     (if (criterio (car libros))

         (cons (car libros) (filtro-libros (cdr libros) criterio))

         (filtro-libros (cdr libros) criterio))))

; i) precio mínimo (3 ptos)
```

```
;; Retornar todos los libros cuyo precio sea mayor al precio mínimo
(define (criterio-precio-min min)
         (lambda (libro) (>= (get-precio-minimo libro) min)))


; ii) precio máximo (3 pts)
(define (criterio-precio-max max)
         (lambda (libro) (<= (get-precio-minimo libro) max)))


;;; iii) numero de paginas (3 pts)
;; Acá queda a elección del estudiante como quiere filtrar.
;; Puede ser: filtrar los libros que tengan una cantidad menor, o mayor, o igual
(define (criterio-max-numero-paginas numero-paginas)
         (lambda (libro) (<= (get-numero-paginas libro) max)))
;;; iv) un valor entre el precio mínimo y el precio maximo (4 pts)
(define (criterio-precio-min-max min max)
         (lambda (libro)    (and (<= (get-precio-minimo libro) max)
                                 (>= (get-precio-minimo libro) min))
    )))
```