Segundo parcial de fundamentos de programación

Ángela Villota Gómez

Diciembre 11 de 2007

Nombre:_____ Código: _____

1. Funciones Similares [15 pts]

Escriba una función llamada *general* que sea genérica y que cumpla con el propósito de las dos siguientes funciones:

```
;; sum: (lista de numeros)->numero
;; para calcular la suma de los
;; elementos de una lista
(define (sum alon)
  (cond
   [(empty? alon) 0]
   [else
     (+ (first alon) (sum (rest alon)))
   ]))
```

```
;; prod: (lista de numeros)-> numero
;; para calcular el producto de los
;; elementos de una lista
(define (prod alon)
  (cond
    [(empty? alon) 1]
    [else
    (* (first alon) (prod (rest alon)))
    ]))
```

1. Escriba las diferencias en las funciones:
   1._____
   2._____

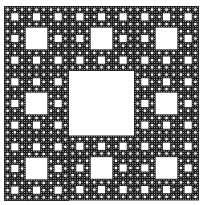
2. Responda: ¿Qué tipo de diferencias son?
   1._____
   2._____

3. Responda: ¿Qué estrategia usaría para escribir una función genérica dadas las diferencias anteriores?
   _____
   _____

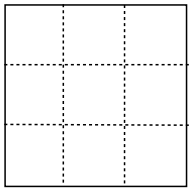
4. Escriba la función *general*:
   _____
   _____
   _____
   _____
   _____
   _____

2. Recursión Generativa [40 pts]

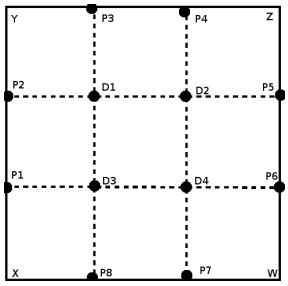
El Problema: la alfombra de Sierpinski



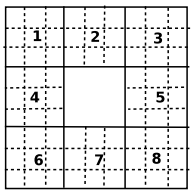
La alfombra de Sierpinski es un fractal que está hecho usando cuadrados (similar al triangulo que vimos en clase). Los pasos para dibujarlos son los siguientes:
Paso No. 1: se calcula $\frac{1}{3}$ del lado del cuadrado



Paso No. 2: se encuentran todos los puntos que definen los cuadrados internos



Paso No. 3: se hace el llamado recursivo con todos los cuadrados menos el del centro, hasta que el cuadrado sea muy pequeño. Cuando un cuadrado es muy pequeño, no hay que partirlo, hay que dibujarlo.



En los pasos anteriores se puede notar que usando la estrategia: *divide y vencerás* se puede encontrar una solución

2.1. Preguntas

2.1.1. Entendiendo el problema [10 pts]

1. Responda: ¿En qué caso el problema es trivial?

   _____

2. Responda: ¿Cuál es la respuesta del programa en el caso trivial?

   _____

3. Responda: ¿Cómo se parte el problema en subproblemas?

   _____

   _____

2.1.2. Funciones auxiliares [20 pts]

Suponga que ya existen las siguientes funciones:

distancia: posn posn ->numero toma dos puntos representados por medio de estructuras posn (como en el paquete de dibujo) y retorna la distancia entre ellos.

dibujar: posnX posnY posnZ posnW ->true dibuja en el lienzo (canvas) el cuadrado delimitado por X, Y, Z y W (ver dibujo del paso 2).

es-pequeño? posnX posnY posnZ posnW → booleano retorna verdadero si el lado del cuadrado es menor de 5 pixeles, en el caso contrario retorna falso.

1. Escriba en scheme la función *tercio* que tiene como entrada dos puntos X y Y, la salida de la función es la tercera parte de la distancia entre X y Y.

2. Escriba en scheme la función *trasladar-en-x* que tiene dos entradas: un punto (posn) y un número que puede ser positivo o negativo. La salida de la función es el punto trasladado en el eje X hacia la derecha si el número es positivo, o a la izquierda si el número es negativo.

3. Escriba en scheme la función *trasladar-en-y* que tiene dos entradas: un punto (posn) y un número que puede ser positivo o negativo. La salida de la función es el punto trasladado en el eje Y hacia abajo si el número es positivo, o hacia arriba si el número es negativo.

2.1.3. Escribiendo el programa [10 pts]

Complete la plantilla que encuentra a continuación con las funciones auxiliares del punto anterior.

```
alfombra: posn posn posn posn -> true
propósito: dibujar la alfombra de sierpinski
(define (alfombra x y z w)
  (cond
    [_______________________________________
     ___________________________________]
    [else
    (local
     ( (define P1 ____________________)
       (define P2 ____________________)
       (define P3 ____________________)
       (define P4 ____________________)
       (define P5 ____________________)
       (define P6 ____________________)
       (define P7 ____________________)
       (define P8 ____________________)
       (define D1 ____________________)
       (define D2 ____________________)
       (define D3 ____________________)
       (define D4 ____________________))
     (and
       (alfombra ______________________________
         ___________________________________ )
       (alfombra ______________________________
         ___________________________________ )
       (alfombra ______________________________
         ___________________________________ )
       (alfombra ______________________________
         ___________________________________ )
       (alfombra ______________________________
         ___________________________________ )
       (alfombra ______________________________
         ___________________________________ )
       (alfombra ______________________________
         ___________________________________ )
       (alfombra ______________________________
         ___________________________________ )
      ))]))
```

3. Problemas con acumuladores, cambio de estado y búsqueda [45 pts]

El Problema

El café internet **n@vegar.com** le solicita a usted que diseñe e implemente una aplicación en scheme para llevar el control de usuarios de sus computadores. El dueño cuenta con 2 computadores que alquila por un periodo de una hora, el valor del alquiler (por una hora) es de $2000. El programa que usted debe escribir debe hacer lo siguiente:

- Ingresar la información relacionada con los alquileres. Los datos que se van a almacenar por cada alquiler son: hora de entrada, hora de salida, nombre del cliente y total a pagar por el alquiler (el total lo debe calcular el programa).
- Para ingresar un cliente, se debe verificar que el computador asignado está disponible.
- El programa debe permitir calcular las ganancias del día, esto es el total de dinero que los clientes pagaron por el alquler de los dos computadores.

3.1. Preguntas

3.1.1. Definición de datos [5 pts]

La información relacionada con un alquiler se almacenará en una estructura; cada vez que se alquila un computador se guardan los datos del alquiler en una lista asociada al computador, es decir una lista que contiene estructuras.

1. Escriba la definición de una estructura llamada alquiler que tenga los campos horaI, horaS, cliente, total.

   ---

2. Escriba la definición de una variable de tipo lista llamada PC1, inicialmente la lista está vacía.

   ---

3. Escriba la definición de una variable de tipo lista llamada PC2, inicialmente la lista está vacía.

3.1.2. Ingresando Datos [30 pts]

La función que permite ingresar los datos se llama *agregar-cliente*, tiene como entrada los datos del alquiler (menos el total) y el código del computador (1 o 2) que se va a alquilar.

1. [5*pts*] Escriba en scheme la función *calcular* que toma dos entradas: la hora de ingreso y la hora de salida. La función calcula el monto que debe pagar un cliente. Recuerde: el precio del alquiler por una hora es $2000, el alquiler se hace mínimo por una hora, La hora debe ser un entero.

2. [5*pts*] Escriba en scheme la función *agregar* que tiene dos entradas un elemento y una lista, la función retorna una lista que contiene el nuevo elemento (más los anteriores).

3. [10*pts*] Complete la función *disponible?* que toma como entrada la hora de entrada de un cliente y una lista (asociada a un computador). La salida de la función es verdadero si el computador está ocupado a la hora de entrada requerida, falso en el caso contrario.

```
; disponible?: numero numero -> booleano
; propósito: determinar si un computador
; está disponible a una hora dada.
; Análisis de casos:
; 1. Si la lista está vacía está disponible
; 2. Si la hora está entre  la hora de
     inicio y la salida de un alquiler
     entonces no está disponible
; 3. de lo contrario seguir buscando

(define (disponible? hora pc)
  (cond
   [(empty? pc) _________]
   [(and
      (__ (alquiler-horaI (first pc)) hora)
      (__ (alquiler-horaS (first pc)) hora))
       ________]
   [else
    ________________________]))
```

4. [10*pts*] Complete la función *agregar-cliente*

```
; agregar-cliente: num num sím num-> void
; propósito: agregar la información de un
; cliente a la lista de alquileres del
; computador indicado por pc
; efecto: la lista PC1 o PC2 tienen un
; nuevo elemento
; Análisis de casos:
 1. Si pc es 1, agrego a PC1
 2. Si pc es 2 agrego a PC2

(define (agregar-cliente ent sal cli pc)
```

```
(cond
 [(= pc 1)
   (_____
    [(disponible? hora ___)
     ------------------------------------
     ------------------------------------
     ------------------------------------]
    [else
       "El pc 1 está ocupado"])]
 [(= pc 2)
   (_____
    [(disponible? hora ___)
     ------------------------------------
     ------------------------------------
     ------------------------------------]
    [else
       "El pc 2 está ocupado"])]))
```

3.1.3. Calculando las ganancias [10 pts]

1. [5*pts*] Complete función *calcular_total* que tiene como entrada una lista y un número (acumulador). La función retorna la suma de los montos pagados por los clientes al realizar un alquiler.

```
; calcular_total: lista-de-alquileres num ->num
; propósito: calcular la ganancia total sumando
; el monto de cada uno de los alquileres

(define (calcular_total lista acc)
  (cond
    [(empty? ____) _______]
    [else
      (calcular_total (rest lista)
                      ______________________)]
    ))
```

2. [5*pts*] Complete la función *ganancia* que calcula las ganancias obtenidas por el alquiler de los dos computadores.

```
; ganancia:  -> numero
; propósito: calcular la ganancia total, sumando
  la ganancia por alquilar el PC1 y el PC2
(define (ganancia )
  ----------------------------------------)
```