Lenguajes de Programación

Práctica 2

Karla Ramírez Pulido Manuel Soto Romero Alejandro Hernández Mora
Alma Rocío Sánchez Salgado Silvia Díaz Gómez

Semestre 2021-1
Facultad de Ciencias, UNAM

Fecha de inicio: 14 de octubre de 2020
Fecha de entrega: 21 de octubre de 2020

1. Objetivos

Aprender a utilizar funciones anónimas, *map, foldl, foldr, filter, take, drop* y definir tipos de datos abstractos.

2. Ejercicios

Programa las siguientes funciones en el dialecto de *Racket, PLAI*. [1].

A continuación se presentan las firmas de las funciones y un ejemplo de uso de cada una de ellas, asegúrate de hacer las pruebas unitarias correspondientes.

1. Una función que reciba un número n, un número r, y devuelva el conjunto de los primeros r múltiplos de n (empezando por n). En caso de necesitar alguna función auxiliar para este ejercicio, ésta deberá ser anónima. Puedes usar listas por comprensión.

```
;; multiplos: number number -> (listof number)
(define (multiplos n r) ... )

> (multiplos 73 8)
'(73 146 219 292 365 438 511 584)
```

2. El predicado `divisor?` que reciba un número n y un número m y devuelva verdadero si m divide a n, falso en otro caso. Si m es cero, lanza un error.

```
;; divisor?: number number -> boolean
(define (divisor? m n) ... )

>(divisor? 0 3)
Error: El cero no es divisor de nadie
> (divisor? 5 30)
#t
```

[1] Está prohibido utilizar funciones primitivas del lenguaje que resuelvan directamente los ejercicios.

3. Función que recibe un número n como parámetro y devuelve el conjunto de divisores de n.

```
;; divisores: number -> (listof number)
(define (divisores n) ... )

> (divisores 135)
'(1 3 5 9 15 27 45 135)
```

4. Un predicado que reciba un elemento e y una lista l y decida si e pertenece a l.

```
;; pertenece?: a (listof a) -> boolean
(define (pertenece? e l) ... )

> (pertenece? 3 (list 4 7 8 9 32 4 1 3))
#t
```

5. Una función que reciba una lista l con elementos y devuelva una lista de elementos sin repetir de la lista original l.

```
;; eliminaRepetidos: (listof a) -> (listof a)
(define (eliminaRepetidos lista) ...)

> (eliminaRepetidos (list 1 2 3 5 6 7 8 4 4 1 6 7 8))
'(1 2 3 5 6 7 8 4)
```

Considera la siguiente definición del tipo de abstracto de datos para crear puntos en el plano. Servirá para modelar figuras geométricas.

```
(define-type Punto
[Punto (x number?) (y number?)])
```

6. Una función que reciba dos puntos $p = (x_1, y_1)$ y $q = (x_2, y_2)$; y calcule el punto medio entre p y q. Si alguno de los dos argumentos no es un punto, regresa un error. La fórmula para calcular el punto medio es $p_{medio} = (\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2})$

```
;; punto-medio: Punto Punto -> number
(define (punto-medio p q) ... )

>  (punto-medio (Punto 2 2) (Punto 2 8))
(Punto 2 5)
```

7. Una función que reciba dos puntos $p = (x_1, y_1)$ y $q = (x_2, y_2)$; y calcule la distancia entre p y q. Si alguno de los dos argumentos no es un punto, arroja un error. El cálculo de la distancia entre dos puntos está dado por la fórmula: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

```
distancia: Punto Punto -> number
(define (distancia p q) ... )

>  (distancia (Punto 2 2) (Punto 2 8))
6
```

8. Define un tipo abstracto de datos para crear figuras geométricas. Considera un constructor para:

   - Círculos: Recibe como parámetro un punto, que representa el centro del círculo y un número mayor que cero, que representa el radio.
   - Triángulos: Recibe como parámetro los tres puntos del triángulo.
   - Cuadrados: Recibe como parámetro un punto, que representa el de la esquina superior izquierda del cuadrado, y un número positivo que representa la longitud del lado del cuadrado.
   - Rectángulos: Recibe como parámetro un punto, que representa el de la esquina superior izquierda del rectángulo y dos números positivos que representan el largo y la altura de la base del rectángulo, respectivamente[2].

```
;; Definición del tipo abstracto de datos Figura
(define-type Figura
[Circulo    ...]
[Triangulo  ...]
[Cuadrado   ...]
[Rectangulo ...])
```

9. Una función que reciba una figura y calcule el perímetro de la misma.

```
;; perimetro: Figura -> number
(define (perimetro fig) ... )

>(perimetro (Triangulo (Punto 2 2) (Punto 3 6) (Punto 1 9)))
14.799724712947125
```

10. Una función que reciba una figura y calcule el área de la misma.

```
;; area: Figura -> number
(define (area fig) ... )

>(area (Triangulo (Punto 2 2) (Punto 3 6) (Punto 1 9)))
5.4999999999999964
```

[2]Para todas las figuras geométricas, los parámetros deben recibirse en ese orden y debes respetar el nombre de los constructores.

11. **Punto extra:** Una función que reciba una lista l como parámetro y devuelva el elemento con mayor número de repeticiones en la lista l. Si hay dos o más elementos repetidos el mismo número de veces, regresa el primero de éstos en aparecer de izquierda a derecha en la lista original. Si la lista está vacía, lanza un error.

```
;; masRepetido (listof a) -> a
;; (define (masRepetido lista) ... )

> (masRepetido (list 1 2 3 3 6 6 7 7))
3
> (masRepetido (list 1 2 3 4))
1
> (masRepetido (list 1 2 3 4 5 6 3))
3
```

3. Requerimientos

Deberás entregar tu práctica en equipos de mínimo dos personas y máximo tres. La entrega es a través de la plataforma **Google classroom** antes de las 23:59 del día de entrega indicado, tal y como lo indican los lineamientos de entrega. Deberás adjuntar tu código en un archivo llamado ***Practica2.rkt***. Revisa bien mayúsculas, minúsculas y espacios para el nombre del archivo que se pide.

El orden en el que aparezcan las funciones en el archivo solicitado, debe ser el orden especificado en este PDF, de lo contrario podrán penalizarse algunos puntos. Puedes utilizar funciones auxiliares, se recomienda definirla después de la función que la va a utilizar.

No dudes en aclarar tus dudas en la sesión de laboratorio y vía correo electrónico.

¡Que tengas éxito en la práctica!.