Unidad 6: Introducción a Programación Funcional en Scheme

Ejercicio 1: Funcion tres

Escribir una función llamada "tres" que, dado cualquier valor, devuelve el número 3. **Función**

```
(define (tres x) 3)
```

Consulta

```
(tres 5)
Output: 3
```

Ejercicio 2: Función triple

Escribir una función llamada triple que, dado cualquier valor x, devuelve el triple de x. **Función**

```
(define (triple x)
  (* x 3)
)
```

Consulta

```
(triple 5)
output: 15
```

Ejercicio 3: Mitad de un número

Función que, dado cualquier número, devuelve la mitad de este. **Función**

```
(define (mitad x)
  (/ x 2)
)
```

Consulta

```
(mitad 10)
output: 5
```

Ejercicio 4: duplo de la suma

Escribir una función que, dado cualquier valor x, devuelve el duplo de la suma de 15 + x. **Función**

```
(define (duplo x)
  (* 2 (+ 15 x))
)
```

Consulta

```
(duplo 10)
output: 50
```

Ejercicio 5: Circunferencia

Definir "circunferencia", que dado el radio retorne la longitud de esta.
Función

```
(require math)
(define (circunferencia x)
  (* pi(* x 2))
  )
```

Consulta

```
(circunferencia 5)
Output: 31.41592653589793
```

Ejercicio 6: Celsius a Fahrenheit

Escribir una función que, dada una temperatura en grados Celsius, la devuelva expresada en grados Fahrenheit. La fórmula de conversión es f = (9/5)c + 32. **Función**

```
(define (temperatura x)
  (+ 32 (* 1.8 x))
)
```

Consulta

```
(temperatura 12)
Output: 53.6
```

Ejercicio 7:

Sabiendo que: 1 pie = 12 pulgadas; 1 yarda = 3 pies; 1 pulgada = 2.54 centímetros. Definir tres funciones (yardas, pulgadas y pies), que, dado un valor en centímetros, devuelva esa longitud expresada en esas unidades. **Función**

```
(define (pulgadas x)(/ x 2.54))
(define (pies pulgada) (/ pulgada 12))
(define (yarda pies) (/ pies 3))
```

Consulta

```
; Ejemplo de uso:
(define (convertir-centimetros-a-todo x)
  (let* ((pulgadas-resultado (pulgadas x))
         (pies-resultado (pies pulgadas-resultado))
         (yardas-resultado (yarda pies-resultado)))
    (list pulgadas-resultado pies-resultado yardas-resultado)))

; Convierte 100 centímetros a pulgadas, pies y yardas:
(convertir-centimetros-a-todo 100)
Output:
'(39.37007874015748 3.2808398950131235 1.0936132983377078)
```

Ejercicio 8: Superficie

Definir "superficie", una función que dado el ancho y el largo de una habitación retorne su superficie. **Función**

```
(define (superficie x y)
  (* x y)
)
```

Consulta

```
(superficie 10 10)
Output: 100
```

Ejercicio 9: Base y altura de un triángulo

Definir una función para ingresarle la base y la altura de un triángulo, y devuelva el valor del área. **Función**

```
(areaTriangulo 12 15)
```

Consulta

```
(areaTriangulo 12 15)
Output: 90.0
```

Ejercicio 10: Hipotenusa

La relación entre los lados (a,b) de un triángulo y la hipotenusa viene dada por la fórmula `a2 + b2 = h2`. Definir una función para que, dadas las longitudes de los lados, calcule y devuelva la hipotenusa. **Función**

```
(define (hipotenusa a b)
  (sqrt(+ (* a a)(* b b)))
)
```

Consulta

```
(hipotenusa 5 5)
Output: 7.0710678118654755
```

Ejercicio 11: Área de un triángulo

El área de un triángulo cuyos lados son a, b y c se puede calcular por la fórmula: A = sqrt(p * (p - a) * (p - b) * (p - c)) donde p = (a+b+c)/2. Definir una función para ingresarle a, b y c, y devuelva el área del triángulo. **Función**

```
(define (area-triangulo a b c)
  (let* ((p (/ (+ a b c) 2)) ; semiperímetro
         (area (sqrt (* p (- p a) (- p b) (- p c))))) area))
```

Consulta

```
# Consulta
(area-triangulo 3 4 5)
Output: 6
```

Ejercicio 12: Ecuación de primer grado

Definir una función que acepte los coeficientes a, b y c de la ecuación de primer grado a * x + b = c, y devuelva el valor de la raíz. **Función**

```
(define (ecuacion-primer-grado a b c)
  (/ (- c b)a )
)
```

Consulta

```
(ecuacion-primer-grado 10 2 5)
Output: 1/10
```

Ejercicio 13: Cociente y resto de la división de dos numeros enteros

Definir una función que acepte dos números enteros, y devuelva una lista con el cociente y el resto la división entera entre los dos números.

Función con lista

```
(define (cociente-residuo x y)
  (list (quotient x y) (remainder x y))
)
```

Consulta

```
(cociente-residuo 10 2)
Output: '(5 0)
```

Función sin lista (solo print)

```
(define (cociente-residuo x y)
  (display "Cociente: ")
  (display (quotient x y))
  (newline)                ; Salto de línea
  (display "Resto: ")
  (display (remainder x y))
)
```

Consulta

```
(cociente-residuo 10 2)
Output:
  Cociente: 5
  Resto: 0
```

Ejercicio 14: Segundos, minutos y horas.

Definir una función que, dada una cantidad de segundos, devuelva una lista con la misma cantidad expresada en minutos y segundos. **Función**

```
(define (minutos segundos)
  (list
    (quotient segundos 60) (remainder segundos 60)
  )
)
```

Consulta

```
(minutos 130)
Output:
'(2 10) => 2 minutos 10 segundos
```

Ejercicio 15: Porcentaje de varones y mujeres

Definir una función que acepte la cantidad de varones y mujeres que hay en un recinto, y devuelva una lista con el porcentaje de varones y mujeres. **Función**

```
(define (total x y)
  (+ x y))
; ------------------------------------------------ ;
(define (porcentaje-varones-mujeres varones mujeres)
  (let ((total (total varones mujeres)))
    (list
      (/ (* varones 100.0) total)
      (/ (* mujeres 100.0) total)
    )
  )
)
```

Consulta

```
(porcentaje-varones-mujeres 55 45)
Output: '(55.0 45.0)
; ---------------------------------------- ;
(porcentaje-varones-mujeres 34 21)
Output: '(61.81818181818182 38.18181818181818)
```

Ejercicio 16:

Utilizando las expresiones descriptas anteriormente, definir la función pasaje, que reciba una medida en centímetros, y retorne una lista con esa medida expresada en pulgadas, pies y yardas. **Función** `let` permite crear asignaciones locales de expresiones a identificadores.

```
(define (pasaje cm)
  (let (
        (pulgadas (/ cm 2.54))
        (pies (/ cm 30.48))
        (yardas (/ cm 91.44))
        )
    (list pulgadas pies yardas)
  )
)
```

Consulta

```
(pasaje 100000)
Output: '(39370.07874015748 3280.839895013123 1093.6132983377079)
```

Ejercicio 17: Doble

Definir una función llamada doblar que, dada una lista de 3 números, devuelve el doble de su primera componente. **Función**

```
(define (doble-primer-elemento lista)
  (if (null? lista)
    0
    (* (car lista) 2)
  )
)
```

Consulta

```
(doble-primer-elemento '(13 10 23))
Output: 26
```

Ejercicio 18: Producto por tres

Definir una función llamada producto_por_tres que dada una lista de 3 números devuelva otra lista con los mismos multiplicados por 3. **Función**

```
(define (producto_por_tres lista)
  (if (null? lista)
```

```
        lista
        (cons (* (car lista) 3) (producto_por_tres (cdr lista) ) )
    )
)
```

Consulta

```
(producto_por_tres '(2 4 6))
Output: '(6 12 18)
```

Función Alternativa pasando 2 listas

```
 (define (producto_por_tres lista1 lista2)
  (if (null? lista1)
      lista2
      (cons (* (car lista1) 3) (producto_por_tres (cdr lista1) lista2) )
  )
)
```

Consulta

```
(producto_por_tres3 '(2 4 6) '())
Output: '(6 12 18)
```