Práctica 1: Programación funcional en Scheme

Entrega de la práctica

Para entregar la práctica debes subir a Moodle el fichero `practica01.rkt` con una cabecera inicial con tu nombre y apellidos, y las soluciones de cada ejercicio separadas por comentarios. Cada solución debe incluir:

- La **definición de las funciones** que resuelven el ejercicio.
- Un conjunto de **pruebas** que comprueben su funcionamiento, y el de funciones auxiliares que hayamos definido, utilizando el API RackUnit.

Por ejemplo, supongamos que el primer ejercicio de la práctica 1 sea implementar la función `suma-cuadrados` que recibe dos números y devuelve la suma de sus cuadrados y se proponen en el enunciado los siguientes ejemplos:

```
1  (suma-cuadrados 10 10) ; ⇒ 200
2  (suma-cuadrados -2 9) ; ⇒  85
```

La solución se debería entregar de la siguiente forma:

`practica01.rkt` :

```
1  ;; José Fernandez Muñoz
2
3  #lang racket
4  (require rackunit)
5
6  ;;
7  ;; Ejercicio 1: suma-cuadrados
8  ;;
9
10 ;; Función auxiliar cuadrado
11
```

```
(define (cuadrado x)
    (* x x))

(check-equal? (cuadrado 2) 4)
(check-equal? (cuadrado 10) 100)
(check-equal? (cuadrado -2) 4)

;; Función principal suma-cuadrados

(define (suma-cuadrados x y)
    (+ (cuadrado x) (cuadrado y)))

(check-equal?  (suma-cuadrados 10 10)  200)
(check-equal?  (suma-cuadrados -2 9)  85)

;;
;; Ejercicio 2:
;;

...
```

En los casos de prueba se deben incluir los ejemplos del enunciado del ejercicio.

Ejercicios

Ejercicio 1

a) Implementa la función `(binario-a-decimal b3 b2 b1 b0)` que reciba 4 bits que representan un número en binario y devuelva el número decimal equivalente.

```
(binario-a-decimal 1 1 1 1) ; ⇒ 15
(binario-a-decimal 0 1 1 0) ; ⇒ 6
(binario-a-decimal 0 0 1 0) ; ⇒ 2
```

Nota: recuerda que para realizar esta conversión, se utiliza la siguiente fórmula:

```
n = b3 * 2^3 + b2 * 2^2 + b1 * 2^1 + b0 * 2^0
```

Para la implementación de la expresión debes utilizar la función `expt`.

b) Implementa la función `(binario-a-hexadecimal b3 b2 b1 b0)` que reciba 4 bits de un número representado en binario y devuelva el carácter

correspondiente a su representación en hexadecimal.

```
(binario-a-hexadecimal 1 1 1 1) ; ⇒ #\F
(binario-a-hexadecimal 0 1 1 0) ; ⇒ #\6
(binario-a-hexadecimal 1 0 1 0) ; ⇒ #\A
```

Nota: para realizar esta conversión, como paso intermedio debes pasar primero el número binario a su representación decimal (utilizando la función definida en el apartado anterior) y después a su correspondiente hexadecimal.

Recuerda que la representación hexadecimal de los números decimales del 0 al 9 es el carácter correspondiente a ese número, y que el número decimal 10 se representa con el carácter A, el 11 con el B, y así sucesivamente hasta el 15 que es el F en hexadecimal.

Para la implementación de esta función auxiliar que pasa de decimal a hexadecimal debes usar las funciones `integer->char` y `char->integer`. En la función `char->integer` los caracteres consecutivos están asociados con números consecutivos. Por ejemplo, el entero correspondiente al carácter `#\A` es uno menos que el correspondiente al carácter `#\B`. Los caracteres de números y los de letras no son consecutivos.

Ejercicio 2

Implementa la función `(menor-de-tres n1 n2 n3)` que reciba tres números como argumento y devuelva el menor de los tres, intentando que el número de condiciones sea mínima.

No debes utilizar la función `min`.

Implementa dos versiones de la función:

- versión 1: usando la forma especial `if`
- versión 2 (llámala `menor-de-tres-v2`): definiendo una función auxiliar `(menor x y)` que devuelva el menor de dos números (deberás usar también la forma especial `if` para implementarla) y construyendo la función `menor-de-tres-v2` como una composición de llamadas a esta función auxiliar.

```
(menor-de-tres 2 8 1) ;; ⇒ 1
(menor-de-tres-v2 3 0 3) ;; ⇒ 0
```

Ejercicio 3

a) Supongamos las definiciones

```
(define (f x y)
    (cons x y))

(define (g x)
    (cons 2 x))
```

Realiza la evaluación paso a paso de la siguiente expresión

```
(f (g (+ 2 1)) (+ 1 1))
```

mediante el **modelo de sustitución**, utilizando tanto el **orden aplicativo** y como el **orden normal**.

Escribe la solución entre comentarios en el propio fichero `.rkt` de la práctica.

b) Supongamos las definiciones

```
(define (f x)
    (/ x 0))

(define (g x y)
    (if (= x 0)
        0
        y))
```

Igual que en el apartado anterior, realiza la evaluación paso a paso de la siguiente expresión

```
(g 0 (f 10))
```

mediante el **modelo de sustitución**, utilizando tanto el **orden aplicativo** y como el **orden normal**. Y escribe la solución entre comentarios en el propio fichero `.rkt` de la práctica.

Ejercicio 4

Implementa la función `(tirada-ganadora t1 t2)` que reciba 2 parejas como argumento, donde cada pareja representa una tirada con 2 dados (contiene dos números). La función debe determinar qué tirada es la ganadora, teniendo en cuenta que será aquella cuya suma de sus 2 dados esté más próxima al número 7. La función devolverá 1 si `t1` es la ganadora, 2 si `t2` es la ganadora o bien 0 si hay un empate. Este último caso se producirá cuando la diferencia con 7 de ambas tiradas es la misma.

```
(tirada-ganadora (cons 1 3) (cons 1 6)) ; ⇒ 2
(tirada-ganadora (cons 1 5) (cons 2 2)) ; ⇒ 1
(tirada-ganadora (cons 6 2) (cons 3 3)) ; ⇒ 0
```

Ejercicio 5

Supongamos que queremos programar un juego de cartas. Lo primero que debemos hacer es definir una forma de representar las cartas y funciones que trabajen con esa representación. En este ejercicio vamos a implementar esas funciones.

Representaremos una carta por un símbolo con dos letras: la primera indicará su número o figura y la segunda el palo de la carta.

Por ejemplo:

```
(define tres-de-oros '3O)
(define as-de-copas 'AC)
(define caballo-de-espadas 'CE)
```

Debemos definir la función `carta` que devuelve una pareja con el valor correspondiente a su orden en la baraja española (un número) y el palo de la carta (un símbolo).

```
(carta tres-de-oros) ; ⇒ (3 . Oros)
(carta as-de-copas) ; ⇒ (1 . Copas)
(carta 'RB) ; ⇒ (12 . Bastos)
```

Los valores de las cartas de la baraja española son:

```
A (As) ⇒ 1
S (Sota) ⇒ 10
C (Caballo) ⇒ 11
R (Rey) ⇒ 12
```

Para realizar el ejercicio debes definir en primer lugar las funciones `(obten-palo char)` y `(obten-valor char)` que devuelven el palo y el valor, dado un carácter. Y debes implementar la función `carta` usando estas dos funciones.

```
(obten-palo #\O) ; ⇒ Oros
(obten-palo #\E) ; ⇒ Espadas
(obten-valor #\3) ; ⇒ 3
(obten-valor #\S) ; ⇒ 10
```

Pista

Puedes utilizar las funciones `(symbol->string simbolo)` que convierte un símbolo en una cadena y `(string-ref cadena pos)` que devuelve el carácter de una cadena situado en una determinada posición.

Ejercicio 6

Define la función `tipo-triangulo` que recibe como parámetro las coordenadas en el plano de los vértices de un triángulo representados con parejas. La función devuelve una cadena con el tipo de triángulo correspondiente: equilátero, isósceles o escaleno.

Recuerda que un triángulo equilátero es aquel cuyos tres lados tienen la misma longitud, el isósceles el que tiene dos lados iguales y el escaleno el que todos sus lados son diferentes.

Ejemplos:

```
(tipo-triangulo (cons 1 1) (cons  1 6) (cons 6 1)) ; ⇒
"isósceles"
(tipo-triangulo (cons -2 3) (cons  2 6) (cons 5 3)) ; ⇒
"escaleno"
(tipo-triangulo (cons -3 0) (cons  3 0) (cons 0 5.1961)) ;  ⇒
"equilatero"
```

Nota

Para comparar dos números reales debemos comprobar si la resta entre ambos es menor que una constante `epsilon` que hemos definido. Por ejemplo, `epsilon` puede valer 0.0001.

Puedes usar la siguiente función auxiliar:

```
(define epsilon 0.0001)

(define (iguales-reales? x y)
    (< (abs (- x y)) epsilon))
```

Domingo Gallardo, Cristina Pomares, Antonio Botía, Francisco Martínez