Ejercitación Extra № 0 : Expresiones, Tipos Elementales, y Primeras Funciones

Cátedra Redictado Programación I Septiembre 2020

1. Ejercitación

Ejercicio 1. **Convertir y Evaluar en `Racket`**

Convertir las siguientes expresiones a notación prefija y evaluar el resultado en Racket:

1. 1+2+3+4
2. 1+2+3*4
3. 1*2+3*4
4. (1+2)*3+4
5. (1+2)*(3+4)

Ejercicio 2. **Evaluar en `Racket`**

Evaluar las siguientes expresiones en `Racket`

1. (+ 1 2 3 4)
2. (* 1 2 3 4)
3. (- 5 4 3 2)
4. (- (+ 5 4) 3)
5. (+ (- 5 4) (- 3 2))
6. (* (+ 1 2) (- 3 4) (* 5 (+ 6 7)))
7. (/ (+ 65 4) (- 32 1))
8. (/ (- 65 4) 32)

Ejercicio 3. **Expresiones numéricas y su evaluación**

Verificar el resultado de las siguientes expresiones numéricas en `Racket`. Evalúelas en el intérprete, y luego analice y comente brevemente el resultado obtenido usando una tabla.

1. `(expt 53 53)`
2. `(sqrt -1)`
3. `(* (sqrt -1) (sqrt -1))`
4. `( / 4 6)`
5. `( / 4.0 6)`

Ejercicio 4. **Evaluaciones sobre Booleanos**

Realice una evaluación paso a paso en papel de las siguientes expresiones booleanas. Luego compare su cálculo con en el intérprete de `Racket`.

1. `(not (and #f (or #t (not #t))))`

2. `(not (or #f (or #t ( and #t #f))))`
3. `(and (odd? 1) (odd? 2) (odd? 3))`
4. `(or (even? 1) (even? 2) (even? 3))`
5. `(and (odd? 2) (odd? 1) (odd? 3))`
6. `(or (even? 2) (even? 1) (even? 3))`

EJERCICIO 5. **Predicados**

Probando predicados (expresiones booleanas) en Racket. Evalúe las expresiones y comente brevemente el resultado obtenido.

Predicados sobre tipos

1. `(image? 10)`
2. `(string? "hello world")`
3. `(boolean? "false")`
4. `(boolean? #false)`
5. `(integer? 1.0)`
6. `(integer? 1)`
7. `(exact-integer? 1.0)`
8. `(real? 10)`
9. `(real? (sqrt -1))`
10. `(rational? 2/3)`
11. `(number? “c’est une number”)`
12. `(number? 1)`

Predicados en el campo numérico

1. `(integer? pi)`
2. `(integer? (+ 15 1))`
3. `(exact-integer? 1.0)`
4. `(real? 10)`
5. `(real? (sqrt -1))`
6. `(rational? 2/3)`

Predicados sobre relaciones de orden

1. `(boolean=? #f #f)`
2. `(string=? “hello world” “good bye”)`
3. `(equal? “hello world” “good bye” )`
4. `(string>? “hello world” “good bye”)`
5. `(equal? 6 “media docena”)`
6. `(equal? 6 6)`
7. `(equal? “media docena” “media docena”)`
8. `(equal? (circle 30 “solid” “transparent”) (square 60 “solid” “transparent”))`

9. `(equal? (circle 30 “solid” “blue”) (square 60 “solid” “blue”))`

10. `(equal? (rectangle 20 10 “solid” “green”) (rotate 90 (rectangle 10 20 “solid” “green”)))`

Ejercicio 6. Definición de funciones

Defina funciones que realicen las acciones indicadas a continuación. Para cada definición deberá dar: a) la signatura de la función (lo que recibe y lo que retorna), b) el código `Racket` de la misma, y c) cuatro o cinco ejemplos de aplicación de la misma. Puede tomar de referencia el siguiente ejemplo:

```
;duplica-imagen-der:Image -> Image
;----------------------------------
(define (duplica-imagen-der img)
            (beside img img) )
;----------------------------------
(define IMG1 (rectangle 50 "solid" "yellow"))

(duplica-imagen-der IMG1) ; duplica la imagen 1
(duplica-imagen-der (square 70 "outine" "blue")) ; duplica un cuadrado
(duplica-imagen-der (duplica-imagen-der IMG1)) ; cuatriplica: duplica ↩
   duplica
(duplica-imagen-der (above IMG1 IMG1))
```

1. Dada una imagen, devuelva otra imagen, la cual sea igual a la original pero con la estampa de un circulo rojo en su centro, a modo de sello. El círculo rojo será constante y tendrá un diámetro de 15. Llamar a esta función: **estampa-rojo**

2. Dado un entero no negativo, y un color, dibuje las siguientes imágenes una al lado de la otra: círculo, triángulo, cuadrado y una estrella. Llamar a esta función: **figuras-pegadas** (*Observación:* la función **beside** puede ser de utilidad.)

3. Dado un entero no negativo, y un color, dibuja 5 cuadrados apilados, donde el valor dado se corresponde al lado del primer cuadrado, el segundo cuadrado tendrá el lado original mas 10 unidades, el tercero 10 unidades mas que el segundo, el cuarto 10 mas que el tercero, y así sucesivamente, hasta completar las 5 figura pedidas. Llamar a esta función: **cuadrados-apilados** (*Observación:* la función **above** puede ser de utilidad.)

4. Dada una imagen la función realiza una rotación a 45 grados de esta. Llamar a esta función: **rota-45** (*Observación:* la función **rotate** puede ser de utilidad.)

5. Dada una imagen la función realiza un espejado de la misma, devolviendo la imagen original y a su lado, la imagen rotada a 45 grados. Utilice la función definida en el item anterior. Llamar a esta función: **espejada-45**. Para probar su función defina tres constante con imágenes tomadas de internet, y aplique la función recién definida sobre las mismas.

Ejercicio 7. Pequeños Programas

Dados las siguientes expresiones en Racket, abstraer alguno (uno o mas) de sus valores para poder diseñar funciones sobre estas expresiones. Expliqué con sus palabras el porqué de su decisión. Compare sus definiciones con las de sus compañeros: ¿Distintas definiciones? Perfecto!

Código 1

```
(define TRI (triangle 40 "solid" "blue"))

(beside TRI TRI TRI)
```

Código 2

```
(define TRI (triangle 40 "solid" "blue"))
(define CUAD (square 20 "solid" "red"))

(beside TRI CUAD TRI)
```

Código 3

```
(define ESTR (star 25 "solid" "yellow"))
(define CIR (circle 50 "solid" "green"))

(place-image ESTR
             (/(image-width CIR) 2)
             (/(image-height CIR) 2)
             CIR


 )
```

Código 4

```
(define CUAD2 (square 50 "solid" "green"))

(overlay  CUAD2
          (rotate 25 CUAD2)
          (rotate 45 CUAD2)
          (rotate 65 CUAD2)
          (rotate 85 CUAD2)
          )
```