Taller 1 Racket y recursividad Parte 1
Fundamentos de Lenguajes Programación

Carlos Andres Delgado S
carlos.andres.delgado@correounivalle.edu.co

Marzo de 2023

1. Reglas

- Se permiten grupos de 2 a 4 personas
- Debe usarse recursión para resolver los problemas
- Debe usarse sintaxis de Racket paréntesis en caso de entregar en papel, en computador debe ser funcional en lenguaje Racket.
- Las funciones deben definirse con **lambda**

1.1. Indicadores de logro

1. IL 1.1.1 Construye datos recursivamente utilizando métodos de inducción y gramáticas BNF
2. IL 1.1.2 Utiliza gramáticas BNF para guiar la construcción de programas recursivos

2. Enunciado del taller

De acuerdo a la gramática de listas:

```
<lista> ::= '()
 ::= <elemento> <lista>

<elemento> ::= <numero> | <simbolo> | <lista>
```

1. Diseñar la función **predicado-legendario** esta función recibe una lista de funciones predicado (salida booleana) y una lista de listas de elementos. Esta función retorna una lista de listas que contiene en cada posición las lista filtrada para cada predicado, la definición de la función a construir es:

```
(define predicado-legendario
  (lambda    ....
```

Ejemplo:

```
(predicado-legendario (list number? symbol?
                            (lambda (x) (and (number? x)
                                             (> x 3)))
                            (lambda (x) (and (number? x)
                                             (<= x 4))))
                      '(1 2 3 (a b) (2 4 x y z) 5 6 7 (1 (a b 3) 4)))
```

La salida debe ser:

```
(
    (1 2 3 () (2 4) 5 6 7 (1 (3) 4))
    ((a b) (x y z) ((a b)))
    (() (4) 5 6 7 (() 4))
    (1 2 3 () (2 4) (1 (3) 4))
)
```

2. Diseñar la función **multiplicacion-onfire**, esta toma una lista de listas y un valor entero n, retorna una lista de listas, donde cada posición es la lista original multiplicada entre 1 y n, en caso de que $n \leq 0$ se retorna '()

```
(define multiplicacion-onfire
  (lambda ...
```

```
(multiplicacion-onfire
 '(2 4 4 2 (1 2 (8 9) (9 (3 4) 10) 12) 4 5)
 -3)
;;Salida
()
```

```
 (multiplicacion-onfire
 '(2 4 4 2 (1 2 (8 9) (9 (3 4) 10) 12) 4 5)
 10)
 ;;Salida
((2 4 4 2 (1 2 (8 9) (9 (3 4) 10) 12) 4 5)
 (4 8 8 4 (2 4 (16 18) (18 (6 8) 20) 24) 8 10)
 (6 12 12 6 (3 6 (24 27) (27 (9 12) 30) 36) 12 15)
 (8 16 16 8 (4 8 (32 36) (36 (12 16) 40) 48) 16 20)
 (10 20 20 10 (5 10 (40 45) (45 (15 20) 50) 60) 20 25)
 (12 24 24 12 (6 12 (48 54) (54 (18 24) 60) 72) 24 30)
 (14 28 28 14 (7 14 (56 63) (63 (21 28) 70) 84) 28 35)
 (16 32 32 16 (8 16 (64 72) (72 (24 32) 80) 96) 32 40)
 (18 36 36 18 (9 18 (72 81) (81 (27 36) 90) 108) 36 45)
 (20 40 40 20 (10 20 (80 90) (90 (30 40) 100) 120) 40 50))
```