

## Auxiliar 3

### Fold y más Fold

**Profesores:** Ismael Figueroa y Federico Olmedo

**Auxiliares:** Bastián Corrales y Martín Segur

### P1) Fold en listas de Racket

foldl y foldr son funciones que toman tres argumentos en el caso más básico: una función de dos parámetros, un valor base y una lista de elementos.

```
1 (A B -> B) B ListOf[A] -> B
```

Racket

Estas funciones iteran sobre la lista aplicando recursivamente la función proporcionada. En cada paso, toman un elemento de la lista y el acumulador actual (que inicialmente es el valor base), y producen un nuevo valor que se convierte en el acumulador para la siguiente iteración.

foldl procesa los elementos comenzando desde el principio de la lista (izquierda), mientras que foldr procesa los elementos comenzando desde el final de la lista (derecha).

El proceso continúa hasta que no quedan más elementos en la lista, momento en el cual se devuelve el valor acumulado final como resultado.

Sabiendo todo esto, implemente las siguientes funciones solo usando foldl o foldr.

#### P1.a) Pitatoria

implemente la función pitatoria que reciba una lista de números y retorne la multiplicación de todos los elementos de la lista.

```
1 (pitatoria '(2 3 4)) -> 24
```

Racket

#### P1.b) Invertir

Implemente la función invertir que reciba una lista y retorne una nueva lista con los elementos en orden inverso.

```
1 (invertir '(1 2 3 4)) -> (4 3 2 1)
```

Racket

#### P1.c) Filtrar Pares

Implemente la función filtrar-pares que reciba una lista de números y retorne una nueva lista solo con los números pares.

```
1 (test (filtrar-pares '(1 2 3 4 5 6)) '(2 4 6))
```

Racket

#### P1.d) Eliminar Duplicados

Implemente la función eliminar-duplicados que recibe una lista y retorna la lista pero sin elementos duplicados.

```
1 (test (eliminar-duplicados '(1 2 2 3 3 3 4)) '(1 2 3 4))
```

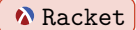
Racket



### P1.e) Sumar múltiplos

Implemente la función `suma-multiplos` que reciba dos listas de números y retorne la suma de todos los múltiplos entre ambas listas.

```
1 (suma-multiplos '(1 2 3) '(4 5 6)) -> (1*4) + (2*5) + (3*6) = 32
```



## P2) Árboles N-arios

Basandonos en la implementación de árboles binarios vistos en clase, trabajaremos con árboles  $n$ -arios.

### P2.a) Definición

Defina el tipo `NTree`

### P2.b) Fold-ntree

Defina `fold-ntree` que abstraiga el patrón recursivo de uso de funciones en árboles  $n$ -arios.

### P2.c) Sum NTree

Defina `sum-n-tree` que calcule la suma total de los valores internos y hojas del árbol.

### P2.d) Contains NTree

Defina `contains-n-tree?` que retorna `#t` si un árbol `nt` contiene el valor `v` en alguno de sus nodos y retorna `#f` en caso contrario.

### P2.e) Select NTree

Defina `select-n-tree` que retorna la lista de valores de un árbol `nt` que cumplen con un predicado `p`.

## P3) (Propuesto) Polinomios


Podemos definir polinomios mediante las siguientes reglas inductivas:

$$\begin{aligned} \langle \text{Pol} \rangle &::= \text{nullp} \\ &\quad | \{ \text{Plus } \langle \text{Int} \rangle \langle \text{Int} \rangle \langle \text{Pol} \rangle \} \end{aligned}$$

Es decir, un `Pol` tiene dos constructores, uno es el polinomio nulo (`nullp`) y el otro sería el no nulo, un (`plus coef grado resto`), donde `coef` es el coeficiente del polinomio, `grado` es el exponente y `resto` es el polinomio que viene después del símbolo `+`

Por ejemplo, el polinomio  $p(x) = 4x^5 + 3x^2 + 5$  se representa mediante:

```
1 (plus 4 5 (plus 3 2 (plus 5 0 (nullp))))
```

 Racket

### P3.a) Definición

Defina el tipo Polinomio

### P3.b) Parse

Defina la función parser que transforme la sintaxis concreta en abstracta. Ojo que los símbolos en la sintaxis concreta están juntos, es decir, `2x5` es UN **SOLO SÍMBOLO** y no hay forma de separarlo fácil con un `match`. Programe funciones auxiliares que a partir del símbolo entero entreguen el coeficiente y el grado.

### P3.c) Fold

Defina `fold-p` que abstraiga el patrón recursivo de uso de funciones.

### P3.d) Eval

Defina `eval` que reciba  $n$  un número entero y  $p$  un polinomio y evalúe  $p(n)$ . Use `fold-p`.