

# Programación Funcional

## Ejercicios de Práctica Nro.6

### Demostraciones I

**Aclaraciones:**

- Los ejercicios siguen un orden de complejidad creciente, y cada uno puede servir a los siguientes. No se recomienda saltar ejercicios sin consultar antes a un docente.
- Recordar que se pueden aprovechar en todo momento las funciones ya definidas, tanto las de esta misma práctica como las de prácticas anteriores.
- Probar todas las implementaciones, al menos en una consola interactiva.
- Es sumamente aconsejable resolver los ejercicios utilizando primordialmente los conceptos y metodologías vistos en clase, dado que los exámenes de la materia evalúan principalmente este aspecto. Para utilizando formas alternativas al resolver los ejercicios consultar a los docentes.

**Ejercicio 1)** Demostrar las siguientes propiedades:

- a. `doble = \x -> 2 * x`
- b. `compose doble doble = cuadruple`

**Ejercicio 2)** Demostrar las siguientes propiedades:

- a. `x && y = not ((not x) || (not y))`
- b. `not (x || y) = not x && not y`

**Ejercicio 3)** Demostrar las siguientes propiedades:

- c. `curry suma' = suma`
- d. `uncurry suma = suma'`

**Ejercicio 4)** Demostrar las siguientes propiedades:

- a. `curry fst = const`
- b. `uncurry (flip const) = snd`

**Ejercicio 5)** Demostrar las siguientes propiedades:

- a. `curry (uncurry f) = f`
- b. `uncurry (curry f') = f'`

**Ejercicio 6)** Dadas las siguientes definiciones

```
assoc :: (a, (b,c)) -> ((a,b), c)
assoc (x, (y,z)) = ((x,y), z)
```

```
appAssoc :: (((a,b), c) -> d) -> (a, (b,c)) -> d
appAssoc f p = f (assoc p)
```

demostrar la siguiente propiedad:

`appAssoc (uncurry (uncurry f)) = uncurry (compose uncurry f)`

**AYUDA:** de necesitar el principio de extensionalidad, considerar usar el argumento en la forma `(x, (y, z))`.

**Ejercicio 7)** Dada la siguiente definición

`(f . g) x = f (g x)`

- a. definir las siguientes funciones utilizando el operador `(.)` y la menor cantidad de parámetros posible:

- i. `cuadruple`
- ii. `doble`
- iii. `twice`
- iv. `many`

- b. demostrar las siguientes propiedades:

- i. `f . g = compose f g`
- ii. `swap . swap = id`
- iii. `f . (g . h) = (f . g) . h`
- iv. `curry . uncurry = id`
- v. `appAssoc f = f . assoc`

- c. demostrar las siguientes propiedades sin utilizar el principio de extensionalidad ni las definiciones de las funciones:

- i. `doble . doble = cuadruple`
- ii. `curry (uncurry (curry f)) = curry f`
- iii. `appAssoc (uncurry (uncurry f))`  
`= (uncurry . uncurry) f . assoc`
- iv. `(uncurry . uncurry) f . assoc`  
`= uncurry (uncurry . f)`