Trabajo Práctico # 1

Programación Funcional

2 de septiembre de 2019

Aclaraciones:

- *Los ejercicios fueron pensados para ser resueltos en el orden en que son presentados. No se saltee ejercicios sin consultar antes a un docente.*
- *Recuerde que puede aprovechar en todo momento las funciones que ha definido, tanto las de esta misma práctica como las de prácticas anteriores.*
- *Pruebe todas sus 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 evaluación principalmente este aspecto. Si se encuentra utilizando formas alternativas al resolver los ejercicios consulte a los docentes.*

Ejercicio 1

Escribir ocho expresiones que denoten al número `4`, diferentes a las ya vistas (`4`, `2+2`, `3+1`, `doble 2`, `doble (1+1)`). Al menos seis deben usar funciones, de las cuales: al menos dos deben usar una expresión lambda, tres deben usar `doble` y una debe usar `cuadruple`.

Ejercicio 2

Mostrar que la expresión `doble (doble 2)` puede reducirse de otras formas que la vista en clase. *Sugerencia: considerar el* `doble` *más externo, en lugar del interno.*

Ejercicio 3

Reducir `cuadruple 2`, y `cuadruple (cuadruple 2)`. ¿Alguna de ellas puede hacerse de más de una forma?

Ejercicio 4

Definir las funciones `triple`, `succ`, `sumarDos`. Comprobar que `twice succ = sumarDos`.

Ejercicio 5

Dar tres ejemplos de pares de expresiones que sean equivalentes entre sí, pero no estén vinculadas por reducción (además de `cuadruple` y `\x → x+x`). Dos de ellas deben no contenter lambdas.

Ejercicio 6

Realizar la reducción completa de `((twice twice) doble) 3`.

Ejercicio 7

Dar expresiones lambda que sean equivalentes a las siguientes expresiones:

1. `triple`

2. `succ`
3. `sumarDos`
4. `twice`
5. `twice twice`

Ejercicio 8

Reescribir las siguientes funciones sin el uso de `let`, `where` o `if-then-else` cuando sea posible.

1. `f x = let (y,z) = (x,x) in y`
2. `greaterThan (x,y) = if x > y then True else False`
3. `f (x,y) = let z = x + y in g (z,y) where g (a,b) = a - b`