Programación Funcional
Ejercicios - Semana 2

Ejercicio 1

Cree una función multimétodo (multimetodo-figuras) que, dada una figura como la que se muestra imprima la leyenda “Soy un []” donde el [] será reemplazado por el atributo :tipo de la figura en cuestión.

(def cuadrado {:nombre "cuadrado" :tipo "figuras.cuadrado"})
(def rectangulo {:nombre "rectangulo" :tipo "figuras.rectangulo"})

Ejercicio 2

Cree una función multimétodo (multimetodo-things) que, dado un parámetro imprima la leyenda “Soy un []” donde el [] será reemplazado por el el nombre del tipo de dato recibido como se detalla a continuación. La idea es crear una función multimétodo que imite a la función de dispatch de Java.

java.lang.String => “Soy un String”
clojure.lang.PersistentVector => “Soy un vector”
Cualquier otro tipo => “Soy un default”

Ejercicio 3

Defina una función `(fmap f coll)` que aplique la función f al contenido de una colección, preservando el tipo de la misma (`map` retorna siempre una lista infinita). La misma debe al menos operar sobre listas, vectores y diccionarios.

```
(fmap inc (list 1 2 3)) => (2 3 4)
(fmap inc [1 2 3 4]) => [2 3 4 5]
(fmap inc {:a 1 :b 2}) => {:a 2 :b 3}
```

Ejercicio 4

Escriba una función (ultimo) que devuelva el último elemento de una secuencia. La misma debe satisfacer los tests que se muestran.

(ultimo [1 2 3 4 5]) => 5
(ultimo '(5 4 3)) => 3
(ultimo ["b" "c" "d"]) => "d"

Ejercicio 5

El siguiente código Java, es funcional? Justifique, considerando desde el punto de vista de su autor y de un usuario.

```
static boolean anyTrue(boolean[] input) {
    boolean result = false;
    for (boolean b: input) {
        result = result || b;
    }
    return result;
}
```