Programación Funcional

Ejercicios de Práctica Nro.7

Inducción/Recursión I

Aclaraciones:

- Los ejercicios siguen un orden de complejidad creciente, y cada uno puede servir a los siguientes. No se recomienda saltear 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.

Sección I

Ejercicio 1) Dar las reglas que definen el conjunto inductivo correspondiente a la parte totalmente definida del tipo algebraico **`Pizza`** definido por:

```
data Pizza = Prepizza | Capa Ingrediente Pizza
data Ingrediente = Aceitunas Int | Anchoas | Cebolla
                 | Jamón         | Queso   | Salsa
```

Ejercicio 2) Dar la formación esquemática de una función definida por recursión estructural sobre el tipo trabajado en el ejercicio anterior.

Ejercicio 3) Dar el tipo de cada una de las siguientes funciones, y definirlas utilizando recursión estructural.

a. **`cantidadDeCapas`**, que describe la cantidad de capas de ingredientes de la misma.
b. **`cantidadDeAceitunas`**, que describe la cantidad de aceitunas que hay en una pizza dada.
c. **`duplicarAceitunas`**, que dada una pizza, describe otra pizza de forma tal que se cumpla la siguiente propiedad:

```
cantidadDeAceitunas (duplicarAceitunas p)
  = 2 * cantidadDeAceitunas p
```

d. **`sinLactosa`**, que indica la pizza resultante de remover todas las capas de queso de una pizza dada.
e. **`aptaIntolerantesLactosa`**, que indica si la pizza dada no tiene queso, o sea se cumple la siguiente propiedad:

si `aptaIntolerantesLactosa p = True`
entonces `p = sinLactosa p`

f. **`agregaAceitunasCorrectamente`**, que indica si en la pizza dada nunca se agrega una cantidad negativa o cero de aceitunas.

g. **conDescripcionMejorada**, que toma una pizza que agrega aceitunas correctamente y describe otra pizza que se construyó con exactamente los mismos ingredientes pero donde no se agregan aceitunas dos veces seguidas.
h. **tiene**, que toma un ingrediente y una pizza e indica si la pizza dada tiene el ingrediente dado.

Ejercicio 4) Demostrar las siguientes propiedades:

a.
```
cantidadDeAceitunas Prepizza
  = cantidadDeAceitunas(conDescripcionMejorada Prepizza)
```

b.
```
cantidadDeAceitunas (Capa Queso Prepizza)
  = cantidadDeAceitunas
      (conDescripcionMejorada (Capa Queso Prepizza))
```

c.
```
cantidadDeAceitunas (Capa (Aceitunas 8)
                          (Capa Queso Prepizza))
  = cantidadDeAceitunas
      (conDescripcionMejorada
         (Capa (Aceitunas 8)
               (Capa Queso Prepizza)))
```

d.
```
cantidadDeAceitunas (Capa (Aceitunas 9)
                          (Capa (Aceitunas 8)
                                (Capa Queso Prepizza)))
  = cantidadDeAceitunas
      (conDescripcionMejorada
         (Capa (Aceitunas 9)
               (Capa (Aceitunas 8)
                     (Capa Queso Prepizza))))
```

CONSEJO: al escribirlo en papel, usar abreviaturas (e.g. **cantAcs** por **cantidadDeAceitunas**, **conDescM** por **conDescripcionMejorada**, **C** por **Capa**, **Ac** por **Aceitunas**, **Q** por **Queso** y **Pp** por **Prepizza**).

Sección II

Ejercicio 1) Dar las reglas que definen el conjunto inductivo correspondiente a la parte totalmente definida de los tipos algebraicos **`Planilla`** y **`Equipo`** dados por las siguientes definiciones:

```
type Nombre = String
data Planilla = Fin | Registro Nombre Planilla
data Equipo = Becario Nombre
            | Investigador Nombre Equipo Equipo Equipo
```

Ejercicio 2) Dar la formación esquemática de una función definida por recursión estructural sobre cada uno de los tipos trabajados en el ejercicio anterior.

Ejercicio 3) Dar el tipo de cada una de las siguientes funciones, y definirlas utilizando recursión estructural.

a. **`largoDePlanilla`**, que describe la cantidad de nombres en una planilla dada.
b. **`esta`**, que toma un nombre y una planilla e indica si en la planilla dada está el nombre dado.
c. **`juntarPlanillas`**, que toma dos planillas y genera una única planilla con los registros de ambas planillas.
d. **`nivelesJerarquicos`**, que describe la cantidad de niveles jerárquicos de un equipo dado.
e. **`cantidadDeIntegrantes`**, que describe la cantidad de integrantes de un equipo dado.
f. **`planillaDeIntegrantes`**, que describe la planilla de integrantes de un equipo dado.

Ejercicio 4) Demostrar las siguientes propiedades:

a. ∀ `p :: Planilla` .
```
largoDePlanilla (juntarPlanillas Fin p)
 = largoDePlanilla Fin + largoDePlanilla p
```

b. ∀ `p :: Planilla` .
```
largoDePlanilla
  (juntarPlanillas (Registro "Edsger" Fin) p)
 = largoDePlanilla (Registro "Edsger" Fin)
     + largoDePlanilla p
```

c. ∀ p :: Planilla .
```
largoDePlanilla
  (juntarPlanillas (Registro "Alan"
                     (Registro "Edsger" Fin)))
                   p)
 = largoDePlanilla (Registro "Alan"
                     (Registro "Edsger" Fin)))
     + largoDePlanilla p
```

d. ∀ p :: Planilla .
```
largoDePlanilla
  (juntarPlanillas (Registro "Alonzo"
                     (Registro "Alan"
                       (Registro "Edsger" Fin)))
                   p)
 = largoDePlanilla (Registro "Alonzo"
                     (Registro "Alan"
                       (Registro "Edsger" Fin)))
     + largoDePlanilla p
```

Ejercicio 5) Demostrar las siguientes propiedades:

a.
```
largoDePlanilla (planillaDeIntegrantes (Becario "Alan"))
  = cantidadDeIntegrantes (Becario "Alan")
```

b.
```
largoDePlanilla (planillaDeIntegrantes (Becario "Brian"))
  = cantidadDeIntegrantes (Becario "Brian")
```

c. ∀ n :: Nombre .
```
largoDePlanilla (planillaDeIntegrantes (Becario n))
  = cantidadDeIntegrantes (Becario n)
```

d.
```
largoDePlanilla
   (planillaIntegrantes
      (Investigador "Alonzo" (Becario "Alan")
                             (Becario "Alfred")
                             (Becario "Stephen")))
 = cantidadIntegrantes
      (Investigador "Alonzo" (Becario "Alan")
                             (Becario "Alfred")
                             (Becario "Stephen"))
```

e.
```
∀ n :: Nombre .
  ∀ n1 :: Nombre . ∀ n2 :: Nombre . ∀ n3 :: Nombre .
largoDePlanilla
   (planillaIntegrantes
      (Investigador n (Becario n1)
                      (Becario n2)
                      (Becario n3)))
  = cantidadIntegrantes
      (Investigador n (Becario n1)
                      (Becario n2)
                      (Becario n3))
```

f.
```
largoDePlanilla
   (planillaIntegrantes
      (Investigador "Oswald"
         (Investigador "Alonzo" (Becario "Alan")
                                (Becario "Alfred")
                                (Becario "Stephen"))
         (Investigador "John"   (Becario "Brian")
                                (Becario "Graham")
                                (Becario "Ioan"))
         (Investigador "Robert" (Becario "Gordon")
                                (Becario "John")
                                (Becario "Raymond"))))
  = cantidadIntegrantes
      (Investigador "Oswald"
         (Investigador "Alonzo" (Becario "Alan")
                                (Becario "Alfred")
                                (Becario "Stephen"))
         (Investigador "John"   (Becario "Brian")
                                (Becario "Graham")
                                (Becario "Ioan"))
         (Investigador "Robert" (Becario "Gordon")
                                (Becario "John")
                                (Becario "Raymond"))))
```

Sección III

Ejercicio 1) Dar las reglas que definen el conjunto inductivo correspondiente a la parte totalmente definida del tipo algebraico `Dungeon a` definido por:

```
data Dungeon a =
          Habitacion a
        | Pasaje (Maybe a) (Dungeon a)
        | Bifurcacion (Maybe a) (Dungeon a) (Dungeon a)
```

Ejercicio 2) Dar la formación esquemática de una función definida por recursión estructural sobre el tipo trabajado en el ejercicio anterior.

Ejercicio 3) Dar el tipo de cada una de las siguientes funciones, y definirlas utilizando recursión estructural.

a. **`cantidadDeBifurcaciones`**, que describe la cantidad de bifurcaciones de un dungeon dado.
b. **`cantidadDePuntosInteresantes`**, que describe la cantidad de puntos interesantes de un dungeon dado. Los puntos interesantes son los lugares donde puede aparecer un elemento.
c. **`cantidadDePuntosVacios`**, que describe la cantidad de puntos interesantes del dungeon dado en las que no hay ningún elemento.
d. **`cantidadDePuntosCon`**, que dado un elemento y un dungeon, describe la cantidad de puntos interesantes del dungeon en las que se encuentra el elemento dado.
e. **`esLineal`**, que indica si no hay bifurcaciones en un dungeon dado.
f. **`llenoDe`**, que dado un elemento y un dungeon, indica si el elemento se encuentra en todas las posiciones del dungeon.

Ejercicio 4) Dada la siguiente definición:

```
data Tesoro = Cofre | Oro | Joyas
```

demostrar las siguientes propiedades:

a.
```
∀ x :: a .
 cantidadDePuntosVacios (Habitacion x) = 0
```

b.
```
cantidadDePuntosVacios (Pasaje Nothing
                            (Habitacion Joyas)) = 1
```

c.
```
∀ y :: a . ∀ x :: a .
 cantidadDePuntosVacios (Pasaje (Just y)
                            (Habitacion x)) = 0
```

d.
```
cantidadDePuntosVacios
  (Bifurcacion Nothing
     (Pasaje Nothing    (Habitacion Joyas))
     (Pasaje (Just Oro) (Habitacion Cofre))) = 2
```

e.
```
∀ z :: a . ∀ y :: a . ∀ x :: a .
cantidadDePuntosVacios
  (Bifurcacion Nothing
     (Pasaje Nothing  (Habitacion z))
     (Pasaje (Just y) (Habitacion x))) = 2
```

f.
```
cantidadDePuntosVacios
  (Bifurcacion (Just Cofre)
     (Bifurcacion Nothing
        (Pasaje Nothing    (Habitacion Joyas))
        (Pasaje (Just Oro) (Habitacion Cofre)))
     (Bifurcacion Nothing
        (Pasaje (Just Oro) (Habitacion Oro)
        (Pasaje Nothing    (Habitacion Joyas)))) = 4
```

Ejercicio 5) Dadas las siguientes definiciones:

```
data VariasCosas a b = Objeto a | Criatura b
data Monstruo = Gargola | Dragon | Troll
```

demostrar las siguientes propiedades:

a.
```
cantidadDePuntosCon (Criatura Troll)
  (Habitacion (Objeto Oro)) = 0
```

b.
```
cantidadDePuntosCon (Criatura Troll)
  (Pasaje (Just (Criatura Troll))
          (Habitacion (Objeto Oro)))) = 1
```

c.
```
cantidadDePuntosCon (Criatura Troll)
  (Pasaje (Just (Criatura Troll))
          (Habitacion (Criatura Troll)))) = 2
```

d.
```
cantidadDePuntosCon (Criatura Troll)
  (Bifurcacion (Just (Criatura Troll))
        (Pasaje (Just (Criatura Troll))
                (Habitacion (Objeto Oro)))
        (Pasaje (Just (Criatura Troll))
                (Habitacion (Criatura Troll)))) = 4
```

e.
```
cantidadDePuntosCon (Criatura Troll)
  (Pasaje (Just (Criatura Troll))
     (Bifurcacion (Just (Criatura Troll))
           (Pasaje (Just (Criatura Troll))
                   (Habitacion (Objeto Oro)))
           (Pasaje (Just (Criatura Troll))
                   (Habitacion (Criatura Troll))))) = 5
```