Paradigmas de Programación **Examen Final** **21/12/2019**

Condiciones de aprobación

Para aprobar es necesario simultáneamente: • completar bien el 60% del examen, y • obtener al menos la mitad de los puntos **en cada paradigma**.	**En todas tus respuestas sé puntual**, no pierdas el foco de lo que se pregunta. Respuestas en exceso generales son tan malas como respuestas incompletas. 

Parte A

Una exportadora agrícola tiene que modificar el sistema de retenciones, que se basa tanto en el cultivo (trigo, soja o maíz) como en la cantidad de hectáreas que siembra, lo que lo convierte en una chacra, hacienda o estancia, considerando de menor a mayor terreno. La solución para consultar la retención, es la siguiente:

```
class Cultivo {}

class Trigo inherits Cultivo {
 var cantidadHectareas
 var retencion
 method retencion() {
   retencion = if (cantidadHectareas.even()) 0.5 else 0.3
   if (cantidadHectareas < 100) /* chacra */                    retencion = retencion + 0.5
   if (cantidadHectareas.between(100, 2000)) /* hacienda */ retencion = retencion + 1.5
   if (cantidadHectareas > 2000) /* estancia */                retencion = retencion + 2.5
   retencion = retencion * cantidadHectareas
 }


class Soja inherits Cultivo {
 var hectareas
 method retencion() {
    var retencion = 1000
    if (hectareas < 100) /* chacra */                    retencion = retencion * 0.5
    if (hectareas.between(100, 2000)) /* hacienda */ retencion = retencion * 1.5
    if (hectareas > 2000) /* estancia */                retencion = retencion * 2.5
    return retencion
  }
 }

  class Maiz inherits Cultivo {
   var cantidadHectareasSembradas
   var cantidadHectareasPerdidas
   var hectareas = cantidadHectareasSembradas - cantidadHectareasPerdidas
   var retencion
   method retencion() {
     retencion = if (cantidadHectareasPerdidas > 100) 0 else 1.10
     if (hectareas < 100) /* chacra */                    retencion = retencion + 0.5
     if (hectareas.between(100, 2000)) /* hacienda */ retencion = retencion + 1.5
     if (hectareas > 2000) /* estancia */                retencion = retencion + 2.5
     retencion = retencion * cantidadHectareasSembradas
   }
  }
```

1. Justifique de acuerdo a lo visto en la cursada cuáles son los errores conceptuales de la solución en
   a. Herencia.
   b. Manejo del estado del sistema.
   c. Polimorfismo.
   d. Abstracciones.
2. Codifique una alternativa que resuelva dichos problemas.

Parte B

Tenemos un predicado toma/2 que relaciona a una persona con aquella bebida que le gusta tomar. La bebida puede ser cerveza, que tiene una variedad, un amargor y un porcentaje de alcohol (0 si es cerveza sin alcohol), o vino, que tiene un tipo y una cantidad de años de añejamiento. El vino siempre tiene alcohol. Y también existen gaseosas varias.

```
toma(juan, coca).
toma(juan, vino(malbec, 3)).
toma(daiana, cerveza(golden, 18, 0)).
toma(gisela, cerveza(ipa, 52, 7)).
toma(edu, cerveza(stout, 28, 6)).
```

```
tieneProblemas(Persona):-
  findall(C,(toma(Persona, cerveza(C,_,A)), A>0),Cs),
  findall(V, toma(Persona, vino(V,_)), Vs),
  findall(T, toma(Persona, T), Ts),
  length(Cs, CCs),
  length(Vs, CVs),
  length(Ts, CTs),
  CTs is CCs + CVs.
```

1. Explique y justifique cuál es el significado de lo que se estaría consultando con el siguiente código:
   `?- tieneProblemas(juan).`
2. Justificar V o F
   a. La solución planteada es poco declarativa.
   b. La solución planteada utiliza polimorfismo.
3. Implemente una solución superadora.

Parte C

Estamos armando un programa en Haskell para una hamburguesería donde se pueden pedir combos de hamburguesa y bebida. El modelo con el que trabajamos es el siguiente:

```
data Combo = Combo {hamburguesa:: [Ingrediente] , bebida:: Bebida }
type Bebida = String
type Ingrediente = String
```

El local trabaja con combos pre-armados, pero necesita permitir que los clientes los alteren en base a sus gustos. En particular es de interés poder agregar ingredientes a la hamburguesa del combo y cambiar la bebida.

1. Dadas las siguientes afirmaciones indicar si son verdaderas o falsas, y justificar conceptualmente.
   a. Se podría resolver el problema de agregar el ingrediente a la hamburguesa del combo con una función
      `agregarIngrediente :: Combo -> Ingrediente -> [Ingrediente]`
   b. Para agregarle panceta a la hamburguesa y también cambiarle la bebida por soda a un combo que conocemos como **combo** se podría hacer:
      ```
      > agregarIngrediente combo "panceta"
      > cambiarBebida "soda" combo
      ```
   c. Sería conveniente que el combo sea el último parámetro de las funciones para agregar un ingrediente y cambiar la bebida, ya que facilitaría el uso de las mismas.
2. En base a tus respuestas anteriores, desarrollá las funciones **agregarIngrediente** y **cambiarBebida** y cómo se usarían para agregarle un ingrediente a la hamburguesa del combo y también cambiarle la bebida.
3. Burger Factory:
   a. Defina un combo básico, con queso y coca.
   b. Desarrolle una función que pueda recibir una lista de modificaciones de combo y las aplique sobre el combo básico.
4. Indicar qué conceptos del paradigma usaste en el punto anterior y en dónde.