Paradigmas de Programación **Examen Final** **18/10/2024**

Condiciones de aprobación

Para aprobar es necesario simultáneamente: • completar 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

Queremos estudiar el comportamiento de compra para un vendedor mayorista de productos generales. Contamos con:

```
data Producto = Prod {
    nombre :: [Char],
    precio :: Float,
    precioCantidad :: Float
}
```

```
queEmpieceConA = (== 'a') . head . nombre
barato = (< 50) . precio

juan = [ queEmpieceConA , barato ]
```

1.
   a. Modificar la función queEmpieceConA para que acepte un carácter como parámetro, en lugar de que la *a* sea fija.
   b. Indicar qué cambiaría en la representación de juan, y qué concepto está relacionado.
   c. Incluir los tipos de juan y todas las funciones.
2. Codificar una función que permita establecer, dada una lista de productos, qué productos elegiría una persona, asumiendo que elige un producto que cumpla todos los requisitos.

Parte B

Se tiene el siguiente código Prolog para un sistema de compras de supermercado, con latas y lácteos:

```
precio(lata(atun, 100, nereida), 70).
precio(lata(atun, 200, nereida), 120).
precio(lacteo(laSerenisima,leche), 15).
precio(lacteo(sancor,leche), 22).
cliente(Cliente):-compro(Cliente, _).
```

```
compro(martina, lata(atun, 100, nereida)).
compro(martina, lacteo(sancor, leche)).
compro(aye, lacteo(sancor, leche)).
```

1. Queremos relacionar un cliente y una marca, se cumple si el cliente compró un producto de esa marca. Codificar el predicado que llamaremos **comproMarca/2**.
2. Se desea saber qué marcas son populares, lo cual se cumple si todos los clientes compraron algún producto de esa marca. Tenemos estas dos soluciones:
   a. `marcaPopu1(Marca):-forall(cliente(Cliente),comproMarca(Cliente,Marca)).`
   b. `marcaPopu2(Marca):-cliente(Cliente),forall(cliente(Cliente),comproMarca(Cliente,Marca)).`

   Ambas tienen problemas, explique por qué justificando su respuesta para cada una.

Parte C

Queremos programar un calendario en donde se pueda averiguarse si una fecha dada está libre. En el calendario ya hay eventos agendados que pueden ser, por ahora, eventos de día completo o recordatorios. La fecha esta libre cuando ninguno de los eventos de día completo ocupa esa fecha, sabiendo que con los recordatorios no hay conflicto.
Se tiene la siguiente solución en Wollok.

```
class Calendario {
  var eventos
  method estaLibre(fecha) =
    eventos.all({evento =>
      evento.esRecordatorio() or (not evento.esRecordatorio() and evento.fecha() != fecha)
    })
}
class EventoDiaCompleto {
  var property fecha
  method esRecordatorio(){ return false }
}

class Recordatorio {
  method esRecordatorio(){ return true }
}
```

1. Dada la solución anterior, responder verdadero o falso y **justificar en todos los casos**.
   a. Es necesario agregar la superclase Evento para que los tipos de evento sean polimórficos.
   b. Es posible agregar un nuevo tipo de evento: los de varios días (en ellos se detalla una lista con todos los días que ocupa) sin cambiar el código de la clase calendario.

2. Proponer una nueva solución (código y diagrama) que resuelva los problemas existentes en el código dado (sin introducir nuevos problemas).