


Condiciones de aprobación

Para aprobar es necesario simultáneamente: <ul style="list-style-type: none"> • 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

Se tiene el siguiente código Prolog para un sistema de compras de supermercado.

<pre> marca(cindor, laSerenisima). marca(latuna, nereida). marca(serenito, laSerenisima). cliente(Cliente):-compro(Cliente, _). </pre>	<pre> compro(martina, latuna). compro(martina, cindor). compro(aye, cindor). compro(aye, serenito). </pre>
--	--

1. Se desea conocer *qué clientes son obsesivos*, lo que ocurre cuando sólo compran productos de una marca. En el ejemplo, sólo aye es obsesiva, porque compra sólo cosas de laSerenisima, mientras que martina compra de laSerenisima y nereida.

Se tienen las siguientes soluciones con problemas. Para cada caso justificar conceptualmente por qué tiene problemas:

- a. `obsesivo(Cliente) :- cliente(Cliente), forall(compro(Cliente,Producto), marca(Producto,Marca)).`
 - b. `obsesivo(Cliente) :- marca(_, Marca), forall(compro(Cliente, Producto), marca(Producto, Marca)).`
 - c. `obsesivo(Cliente) :- marca(Producto, _), forall(compro(Cliente, Producto), marca(Producto, Marca)).`
2. Codificar una solución superadora, correcta conceptualmente, pero que en lugar de usar `forall/2` use `not/1`.

Parte B

Dado el siguiente código Haskell para analizar la información de los productos de un supermercado:

<pre> data Producto = Producto { nombre :: String, marca :: String } </pre>	<pre> deMarca unaMarca productos = filter (== unaMarca) . marca productos todosSeLlaman unNombre = all ((== unNombre) . nombre) primerosQueCumplen criterio = map nombre . take 3 . filter criterio </pre>
---	--

1. Una de las 3 funciones dadas no compila por un error de tipos. Identificar cuál, explicar qué problemas tiene y proponer una alternativa que sí tipe.
2. Indicar el tipo de las otras dos funciones dadas y mostrar un ejemplo de invocación para cada una.
3. Dada una lista infinita de productos, y asumiendo que la función con errores de tipos fue corregida, explicar para cada una de las funciones dadas si existe algún escenario en el cual pueda terminar de evaluarse y si existe alguno en el cual no pueda hacerlo.

Parte C

Queremos programar un calendario en donde se pueda averiguar 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 está 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 una superclase Evento para que los distintos tipos de eventos 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), e incorpore los eventos de varios días.