


## Condiciones de aprobación

Para aprobar es necesario simultáneamente: <ul style="list-style-type: none"> <li>• completar el 60% del examen, y</li> <li>• obtener al menos la mitad de los puntos en cada paradigma.</li> </ul>	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 la gente que se va de vacaciones. Contamos con:

<pre>data Destino = Lugar {   nombre :: [Char],   precio :: Float,   atractivos :: [String] }</pre>	<pre>tieneBoliches destino = ((elem 'boliches') . atractivos) destino barato destino = precio destino &lt; 50 alguien = [tieneBoliches, barato]</pre>
-----------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------

- Necesitamos determinar si en un destino hay algún atractivo en particular, no necesariamente boliches:
  - Definir la función **tieneAtractivo** y explicitar su tipo.
  - Volver a definir **alguien** usando **tieneAtractivo** en vez de **tieneBoliches**, indicando qué concepto del paradigma se aprovecha en esta solución que no se usaba en la definición anterior de **alguien**.
- Se cuenta con una función que permite establecer, dada una lista de destinos turísticos, cuáles elegiría una persona según sus requisitos (por ejemplo **alguien**), asumiendo que elige un destino que cumpla al menos 1.
 

```
destinosElegidos requisitos destinos = filter (f requisitos) destinos
f requisitos destino = any destino requisitos
```

Responder verdadero o falso, justificar y corregir la implementación en caso de ser posible:

- La solución funciona correctamente.
- La solución es poco declarativa.
- La solución es poco expresiva.
- Asumiendo que funciona como se explica, si se la invoca con una lista de destinos infinita no se podrá obtener ninguna respuesta independientemente de cuáles sean los requisitos de la persona.

## Parte B

Dada la siguiente base de conocimiento:

<pre>%genero(Canción, Género). genero(el38, rock). genero(sisters, reggae). genero(muchoPorHacer, rock). genero(tusOjos, reggae). genero(bastara, reggae).</pre>	<pre>%toca(Tema, Banda). toca(el38, divididos). toca(sisters, divididos). toca(muchoPorHacer, riff). toca(tusOjos, losCafres). toca(bastara, losCafres).</pre>
------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------

Y el siguiente predicado, que debería ser cierto cuando una banda únicamente toca rock:

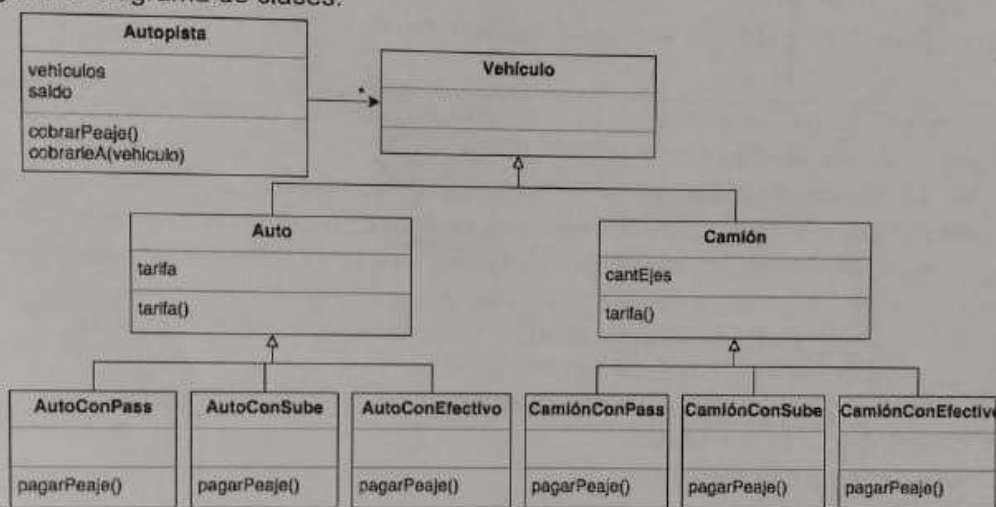
```
rockera(Banda):-
  findall(Tema, (toca(Tema, Banda), genero(Tema, Genero), Genero\=rock), Temas),
  length(Temas, 0).
```

- Decir el resultado de las siguientes consultas:
  - `rockera(riff)`.
  - `rockera(divididos)`.
  - `rockera(42)`.
  - `rockera(Banda)`.
- ¿Qué consulta/s no funciona/n como debería/n? Justificar conceptualmente.
- Escribir una versión más declarativa del predicado `rockera`, solucionando además sus problemas de funcionamiento.

## Parte C

Se sabe que en una autopista se cobra peaje a los vehículos que la usan. Los autos y camiones tienen diferentes formas de calcular la tarifa (en el auto es un valor único y en los camiones según la cantidad de ejes), y además hay vehículos que pagan con telepeaje "Pass", otros con efectivo y otros con la tarjeta Sube. Al efectuarse el cobro suceden dos cosas: el saldo de la empresa que mantiene la autopista aumenta, y además cada vehículo efectúa el pago. Los que pagan con efectivo disminuyen su dinero, los que pagan con sube disminuyen el saldo de la Sube en un 10% más por gasto de servicio (el 10% no va para la autopista), y los que pagan por telepeaje disminuyen en 1 la cantidad de peajes disponibles. La tarifa es \$10 para el auto y \$20 por cada eje en el caso del camión.

Se tiene el siguiente diagrama de clases:



Y suponiendo que el sistema hace lo que debe (funciona) y además esto es parte del código en la clase Autopista:

<pre> method cobrarPeaje(){     vehiculos.forEach{ v =&gt; self.cobrarleA(v) } } </pre>	<pre> method cobrarleA(vehiculo){     saldo = saldo + vehiculo.tarifa()     vehiculo.pagarPeaje() } </pre>
-----------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------

- Indicar verdadero o falso y **justificar** en cada caso con prosa y/o código.
  - Para la solución propuesta, es necesario que exista la clase Vehículo.
  - Hay forma de implementar el resto del código respetando ese diagrama de clases y sin repetir lógica.
  - Si un auto que paga con Sube decide pasarse al sistema Pass de telepeaje no puede hacerlo con este diseño.
  - Si al realizar el cobro a un vehículo el método pagarPeaje tirase error, el sistema quedaría inconsistente.
- Implementar una nueva solución (con código y diagrama) que solucione los problemas encontrados, incluyendo también el código de los vehículos.