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

Se cuenta con la siguiente información:

```
-- Para cada medicamento:
amoxicilina = cura "infección"
bicarbonato = cura "picazón"
ibuprofeno = cura "dolor" . cura "hinchazón"
todosLosMedicamentos = [amoxicilina, bicarbonato, ibuprofeno]

cura sintoma = filter (/= sintoma)

-- Para cada enfermedad / conjunto de síntomas:
malMovimiento = ["dolor"]
varicela = repeat "picazón" [1]

-- Asumimos que existe al menos un medicamento capaz de curar cada enfermedad:
mejorMedicamentoPara sintomas = find (curaTodosLos sintomas) todosLosMedicamentos [2]
curaTodosLos sintomas medicamento = medicamento sintomas == []
```

Se pide:

1. Definir el tipo de un medicamento genérico, y de `curaTodosLos`.
2. ¿Cuáles son los dos conceptos funcionales más importantes aplicados en `mejorMedicamentoPara` y de qué sirvieron?
3. ¿Qué responde esta consulta? Justificar conceptualmente. En caso de errores o comportamientos inesperados, indicar cuáles son y dónde ocurren.

   ```
   > mejorMedicamentoPara varicela
   ```

   Si se reemplaza `todosLosMedicamentos` con lo de abajo, ¿qué respondería la consulta anterior? Justificar conceptualmente.

   ```
   sugestion sintomas = []
   todosLosMedicamentos = [sugestion, amoxicilina, bicarbonato, ibuprofeno]
   ```

Parte B

```
% estaEn(Peli,Personaje)
estaEn(buscandoNemo,nemo).
estaEn(buscandoNemo,dory).
estaEn(quienEnganio,rogerRabbit).
estaEn(quienEnganio,doom).
estaEn(quienEnganio,eddie).
estaEn(naufrago,chuck).
% esAnimado(Personaje)
esAnimado(nemo).
esAnimado(dory).
esAnimado(rogerRabbit).
esAnimado(doom). %Ups, spoiler
```

```
aptaParaAmargos(Peli):-
  findall(Pers,(estaEn(Peli,Pers),esAnimado(Pers)),Ps),
  length(Ps, C), C = 0.

seDivierte(Personaje):-
  findall(Animado,
    (estaEn(Peli1,Personaje),
     estaEn(Peli2,Animado),
     Peli1 = Peli2,
     esAnimado(Animado)),
    Compas),
  length(Compas, C), C >= 1.
```

1. Analizando los predicados aptaParaAmargos y seDivierte

---

[1] `repeat x = x : repeat x`

[2] `find condicion = head . filter condicion`

a. Explicar: ¿Cuándo una peli es apta para amargos? ¿Y cuándo un personaje ~~es buen mimo~~ se divierte? (No cómo lo hacen, sino qué hacen).
b. Dar dos ejemplos de consulta individual para `aptaParaAmargos` y dos para `seDivierte`, una cuya respuesta sea verdadero y una cuya respuesta sea falso, para cada uno. Justifique por cada uno de los casos.
c. ¿Qué responde en cada caso la consulta existencial? **Justificar conceptualmente.**

2. Ambos predicados tienen problemas de declaratividad y de mal uso de pattern matching. Explicar dónde y por qué en cada caso.
3. Realizar una nueva solución arreglando los problemas mencionados en los puntos anteriores.

Parte C

Se desea modelar las líneas telefónicas que pueden contratar los usuarios de una empresa para poder tener internet en sus dispositivos. En una línea **prepaga**, uno puede cargar saldo, que se va agotando con cada kb de internet usado (hay un precio por kb). En una línea con **factura**, uno va acumulando kb de internet usados, y al final de mes se cobra lo que se debe pagar. En una línea **control**, que es como la línea con factura más restrictiva, hay un límite de kb y no permite usar internet si se llega a ese límite. Además, se desea poder conocer un string que muestre el estado de la línea, que incluirá en todos los casos el nombre del plan, el número de teléfono, y además si es línea prepaga el saldo, y si es línea factura ó control los kb usados.
Importante: una línea debe poder cambiar de plan en cualquier momento.
Se tiene la siguiente solución inicial[3]:

```
class Linea {
  var property plan
  var property numero
  var property saldo
  var property precioKb
  var property kbUsados
  var property limiteKb
  method estado(){
    if (plan == "prepago"){
      return "Tu plan es: " + plan + "tu numero es: " + numero + "tu saldo es: " + saldo
    } else {
      return "Tu plan es: " + plan + "tu numero es: " + numero + "usaste kbs: " + kbUsados
    }
  }
  method usarInternet(kbs){
    if (plan == "prepago"){
      if (saldo < kbs * precioKb){
        return "No alcanza el saldo"
      }
      saldo -= kbs * precioKb
    } else if (plan == "factura") {
      kbUsados += kbs
    } else {
      if(limiteKb < kbUsados + kbs){
        return "Se ha superado el límite"
      }
      kbUsados += kbs
} } }
```

1. Responder verdadero o falso y justificar en todos los casos:
   a. Sin modificar los métodos existentes puede agregarse fácilmente un nuevo tipo de plan (por ejemplo, un plan "Ilimitado" que siempre permita usar internet), poniendo un string diferente en la variable `plan`.
   b. Luego de agregar un método `llamarPorWhatsapp`, implementado de esta forma:

```
method llamarPorWhatsapp(linea1,linea2){
   linea1.usarInternet(250)
   linea2.usarInternet(250) }
```

   Si una de las líneas no tiene saldo o si superó el límite de kb (y, por lo tanto, la llamada no puede ocurrir), no se le gasta indebidamente internet a la otra.
2. Proponer una solución superadora para la solución inicial sin repetición de lógica, manteniendo las funcionalidades mencionadas, y respetando los conceptos y buenas prácticas vistos en la materia.

---

[3] ***Nota:*** Suponer que es posible concatenar un string con un número, usando el método suma (+), por lo que no es un problema cuando se usa el + en el método `estado`.