Para aprobar es necesario simultáneamente: ● obtener 8 puntos de 14, y ● obtener al menos la mitad de los puntos en cada paradigma.	Y recordá: 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

1. Se tienen las siguientes declaraciones:

```
funcionMisteriosaV1 = on (++) (take 2)
funcionMisteriosaV2 x y = (on (++) (take 2)) x y
funcionMisteriosaV3 = \x y -> on (++) (take 2) x y
funcionMisteriosaV4 = on (++) . (take 2)
```

```
on f g x y = f (g x) (g y)
```

Algunas de estas variantes de la `funcionMisteriosa` son equivalentes entre sí, mientras que otras no funcionan. Explicá cuáles son equivalentes y por qué, y los errores de las restantes.

2. Se tiene la siguiente definición:

```
iterate f v = v : iterate f (f v)
```

a. La definición es recursiva, pero ¿es correcta? ¿Por qué?
b. Indicá el tipo de las siguientes expresiones, y si al evaluarlas terminan o no.
   - head . iterate (+1)
   - (length . iterate (+5)) 0
   - map (iterate (+5)) [1, 2, 3, 4]

Parte B

Los amigos de ted son profesionales. Sabemos que:

- ted es arquitecto
- lily es maestra y consultora de arte
- marshall es abogado
- robin es periodista
- barney no tiene profesión conocida

1. Escribí la base de conocimiento anterior, de forma que pueda consultarse de esta forma. Justificá las decisiones que tomaste sobre cómo modelar las situaciones de barney y lily.

```
?- profesion(ted, Profesion).
Profesion = arquitecto.
?- profesion(Alguien, arquitecto).
Alguien = ted.
```

2. Se tiene una base de conocimientos con los máximos puntajes históricos de distintos jugadores de básquet:

```
puntos(david, 71).
puntos(kobe, 81).
puntos(michael, 69).
puntos(leBron, 61).
puntos(wilt, 100).
```

Se implementa el siguiente predicado para determinar el máximo goleador entre estos jugadores:

```
goleador(Jugador):-
	findall(Jugador, puntos(Jugador, _), Jugadores),
	member(Jugador, Jugadores),
	puntos(Jugador, Puntos),
	forall((member(OtroJugador, Jugadores), puntos(OtroJugador, OtrosPuntos)),
		OtrosPuntos =< Puntos).
```

¿Se puede resolver sin usar listas? ¿Y sin usar orden superior? Para cada pregunta: en caso afirmativo, indicar cómo (escribir el código); en caso negativo, indicar por qué.

3. Tenemos las siguientes definiciones en Prolog y Haskell para saber si la lista de entrada contiene algún functor bar/2 o tupla de dos elementos que cumpla con tener un 1 en su primer posición.

```
foo(Xs) :- member(bar(1, _), Xs).
foo xs = elem (1, _) xs
```

Una de ellas es incorrecta, mientras que la otra no. Explicar por qué.

Parte C

1. La empresa Cuchuflito S.A. está queriendo implementar un sistema de liquidación de sueldos en Smalltalk. Como paso inicial, entrevistó a dos desarrolladores y les presentó un requerimiento simple: Se tienen empleados que pueden ser gerentes, supervisores u operarios. Los gerentes cobran un sueldo base de $40000, los supervisores $25000 y los operarios $15000. En los 3 casos, se suma un bono por antigüedad de $300 por cada año del empleado en la empresa, dato que se conoce para cualquier empleado. Lo que se necesita es poder consultar el sueldo total de un empleado, por lo que los candidatos deben implementar el método `#Empleado>>sueldoTotal`.

   Pancracio, el primero de los desarrolladores entrevistados, propuso esta solución:

```
#Empleado (V.I.: aniosAntiguedad)
>>sueldoTotal
	^self puesto sueldoBase + self bonoPorAntiguedad
>>bonoPorAntiguedad
	^self aniosAntiguedad * 300
```

```
#Puesto (V.I.: sueldoBase)
```

   Rigoberto, el segundo entrevistado, propuso en cambio esta otra:

```
#Empleado (V.I.: aniosAntiguedad)
>>sueldoTotal
	^self sueldoBase + self bonoPorAntiguedad
>>bonoPorAntiguedad
	^self aniosAntiguedad * 300
```

```
#Gerente (subclase de Empleado)
>>sueldoBase
	^40000
#Supervisor (subclase de Empleado)
>>sueldoBase
	^25000
#Gerente (subclase de Empleado)
>>Operario
	^15000
```

   ¿Cuál de los candidatos te parece que presentó una mejor solución? Justificá.

2. Siguiendo con la entrevista, el próximo requerimiento que se les plantea a los candidatos es el cálculo del sueldo total a pagar por parte de la empresa, teniendo en cuenta a todos sus empleados. Existe esta implementación:

```
#Empresa (V.I.: es)
>>sueldoTotal
    |t|
    t := 0.
    self es do: [:e | t := t + e sueldoTotal].
    ^ t
```

   ¿ Qué responderías en caso de estar <u>en el lugar de cada uno</u> de estos candidatos? En caso de que alguna respuesta sea igual para ambos, indicarlo explícitamente:

   a. ¿Cuál es el beneficio del polimorfismo entre la empresa y el empleado para el cálculo del sueldo total?

   b. ¿Encuentra algún otro lugar donde se utilice polimorfismo en la solución planteada? Considere su implementación del requerimiento anterior.

   c. ¿Cómo mejoraría la solución? Justifique indicando qué conceptos fueron aplicados y dónde/cómo.