


## 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>	<b>En todas tus respuestas sé puntual</b> , no pierdas el foco de lo que se pregunta. Respuestas en exceso generales son tan malas como respuestas incompletas. 
---	---

## Parte A

Se tiene un sistema en el que se realizan compras de productos por internet, con la posibilidad de que exista un envío. Podría no tener envío, o bien ser por Oka (donde hay que adicionar el costo de empaquetado), o bien ser por Correo Argentino (donde el costo del envío es por peso). En general todo costo de envío tiene un costo de comisión por servicio adicional, y debe incluir el IVA (impuesto del 21% sobre el valor total).

Existe el siguiente código Wollok/Smalltalk, que **funciona como se espera**, y calcula el total de una compra:

<pre> class Compra {   var envio   var producto   method totalAPagar(){     if(envio == null) {       return producto.valor()     } else if(envio.esOka()){       return producto.valor() + envio.costo(producto)         + envio.costoEmpaquetado()     } else {       return producto.valor() + envio.costo(producto)     }   } }  class CorreoArgentino {   var valorGramo   var comision   method esOka(){ return false }   method costo(prod){     return (prod.peso() * valorGramo + comision) * 1.21   } }  class Oka {   var costoEnvio   var comision   method esOka(){ return true }   method costo(prod){     return (costoEnvio + comision) * 1.21 }   method costoEmpaquetado(){ return 10 * 1.21 } } </pre>	<pre> #Compra (v.i.: envio, producto) totalAPagar   envio = nil   ifTrue: [^producto valor]   ifFalse: [     envio esOka       ifTrue: [^producto valor +         (envio costo: producto)         + envio costoEmpaquetado]       ifFalse: [^producto valor         + (envio costo: producto)]   ]  #CorreoArgentino (v.i: valorGramo, comision) esOka   ^false costo: prod   ^(prod peso * valorGramo     + comision) * 1.21  #Oka (v.i: costoEnvio, comision) esOka   ^true costo: prod   ^(costoEnvio + comision) * 1.21 costoEmpaquetado   ^10 * 1.21 </pre>
---	--

- Para cada una de las siguientes afirmaciones, responder verdadero o falso y **justificar conceptualmente** en cada caso:
  - Hay un buen uso de polimorfismo porque la compra le manda el mismo mensaje *costo* a cualquier envío, ya sea por Correo Argentino o por Oka.
  - Hay un buen uso de polimorfismo porque la compra le manda el mismo mensaje *esOka* a cualquier envío, ya sea por Correo Argentino o por Oka.
  - Se está rompiendo el encapsulamiento del envío.
  - Las compras sin envío tienen costo de envío 0, entonces es inevitable escribir un "if" para diferenciar los casos con y sin envío.
  - Hay repetición de lógica entre las dos clases de envío y puede solucionarse usando herencia.
- Dar diagrama de clases y código completos de una solución que arregle los problemas descubiertos.

## Parte B

Dada una lista de carreras representados de la siguiente forma:

```

type Distancia = Int
type FactorDificultad = Int

```

```
data Carrera = Carrera { nombre :: String, tramos :: [(Distancia, FactorDificultad)] }
carreras = [ Carrera "Ironman" [(3860, 7), (18000,1), (42200,3)], Carrera "Maratón" [(42200, 3)],
Carrera "Embolsados" [(20, 4)] ]
```

Se solicita desarrollar una función que determine los nombres de aquellas carreras que se pueden completar dada una resistencia límite. Una carrera se puede completar si la suma de las resistencias requeridas de los tramos es menor a la resistencia límite, y la resistencia requerida de un tramo está dada por su distancia multiplicada por el factor de dificultad.

*Ejemplo de uso:* `> puedenCompletarse 150000 carreras`  
`["Maratón", "Embolsados"]`

Se plantean las siguientes soluciones:

```
A) puedenCompletarse x = map nombre . filter (\y -> rt (tramos y) < x)
   rt ts = ( sum . map (\(x,y) -> x * y) ) ts
B) puedenCompletarse _ [] = []
   puedenCompletarse limite (Carrera nombre tramos : carreras)
     | puedeCompletar limite tramos = nombre : puedenCompletarse limite carreras
     | otherwise                    = puedenCompletarse limite carreras
puedeCompletar limite [] = True
puedeCompletar limite ((distancia, factor) : tramos)
  | limite > resistenciaRequerida = puedeCompletar (limite - resistenciaRequerida) tramos
  | otherwise = False
where resistenciaRequerida = distancia * factor
```

Responder Verdadero o Falso y justificar la respuesta:

1. La solución A es más expresiva que la solución B.
2. La solución A es más declarativa que la solución B.
3. La solución A utiliza orden superior, esto puede verse en el uso de `sum`, `map` y `filter`.
4. Dada una lista infinita de carreras "listaLoca", la expresión:

```
head (puedenCompletarse 150000 listaLoca)
```

Termina de evaluarse para la solución A, mientras que para la solución B, al ser recursiva, nunca termina.

## Parte C

Dada la siguiente base de conocimientos:

juega(marcos, futbol). juega(ximena, futbol). juega(carola, futbol). juega(carola, tenis).	campeon(marcos, futbol). campeon(carola, futbol). campeon(carola, tenis).
---	---

Nos piden saber si una persona es un crack. Eso ocurre cuando sale campeón de todos los deportes que juega. Se proponen estas soluciones:

- a. `crack(Persona):- forall(juega(Persona, Deporte), campeon(Persona, Deporte)).`
- b. `crack(Persona):- findall(Campeon, campeon(Persona, Campeon)), Campeonatos,`  
`findall(Juega, juega(Persona, Juega), Jugados),`  
`length(Campeonatos, VecesCampeon), length(Jugados, VecesCampeon).`
- c. `crack(Persona):- not((juega(Persona, Deporte), not(campeon(Persona, Deporte)))).`

Analice las soluciones en términos de

1. Inversibilidad: Incluir al justificar un ejemplo de consulta existencial y su respuesta, y dos de consultas individuales: una cuya respuesta sea verdadera y una cuya respuesta sea falsa.
2. Funcionalidad: ¿Existe algún escenario (no necesariamente con la misma base de conocimientos) ante el cual podría llegarse a respuestas distintas con alguna de las soluciones? Justificar.
3. Al margen de los aspectos anteriores ¿cuál es la mejor solución? Justificar conceptualmente.