


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 desea modelar en el paradigma funcional un sistema de reglas de un firewall. Existen paquetes que llegan al firewall, que tiene una serie de reglas, y el firewall no deja pasar los paquetes que incumplan alguna regla. Un paquete tiene una dirección de origen, una dirección de destino, y un contenido. Y puede haber varias reglas, como por ejemplo una que sólo deja pasar paquetes de direcciones internas (si los primeros 7 caracteres del origen son "192.168"), una que no deja pasar paquetes con cierto destino exacto indicado por el administrador al configurar la regla, o una que no deja pasar paquetes que contengan en su contenido alguna palabra de una lista negra indicada en dicha regla (se puede asumir que existe una función `incluyePalabra :: String -> String -> Bool` que verifica si el primer String contiene al segundo, para simplificar este problema).

Se pide:

1. Definir **tipos de datos y funciones** (explicitando el tipo de todas ellas) para cubrir las necesidades explicadas.
2. Mostrar cómo se representa un firewall de ejemplo que tenga las tres reglas mencionadas anteriormente.
3. Desarrollar una función que permita saber qué paquetes de una lista de paquetes pasan por el firewall.
4. Indicar **dónde y para qué** se utilizaron los siguientes conceptos: composición, aplicación parcial y orden superior.

Parte B

Se conocen los platos que ofrece cada restaurante, y se sabe que se considera bodegón a un restaurante si todos sus platos tienen precio menor a \$300 y además ofrece mila.

| | |
|---|---|
| <pre>% plato(restaurante, plato, precio) plato(laAngioplastia,mila,180). plato(laAngioplastia,bife,230). plato(laAngioplastia,molleja,220). plato(lasVioletas,bife,450). plato(elCuartito,muzza,290).</pre> | <pre>bodegon(Restaurante):- not((plato(Restaurante,_,Precio),Precio >= 300)). bodegon(Restaurante):- tieneMila(Restaurante). tieneMila(Restaurante):- findall(Plato, plato(Restaurante,Plato,_), Platos), member(mila,Platos).</pre> |
|---|---|

1. Responda verdadero o falso y justifique en todos los casos:
 - a. Hay que usar forall para solucionar el error de lógica del predicado bodegon/1.
 - b. El predicado bodegon/1 es inversible.
2. Critique la solución en términos de declaratividad y expresividad.
3. Proponga una solución que resuelva los problemas encontrados en los puntos anteriores.

Parte C

En una estación espacial se sabe que hay distintos roles, que dan cierto prestigio. Se puede **cambiar de rol** a cualquier otro, siempre que se cumpla con sus requisitos (un requisito común a todos los que no están libres es que hay que ser mayor de edad). Si se intenta cambiar de rol por uno que no se puede tener, se considera un error, y debe tratarse como tal. Además, únicamente el rol Mr. Fusión debe registrar un nivel de conocimiento.

Se tiene la siguiente solución:

| | WOLLOK | #Tripulante | SMALLTALK |
|--|--|-------------|---|
| | <pre> class Tripulante { var property rolActual = "Libre" var property edad var property fuerza const property conocimiento method mayorDeEdad() = self.edad() >= 18 method prestigio() = if (self.rolActual() == "Libre") 0 else if (self.rolActual() == "Obrero") 50 else if (self.rolActual() == "MrFusión") 100 else -1 method podesCambiarA(otroRol) = otroRol == "Libre" otroRol == "Obrero" && self.condicionParaObrero() otroRol == "MrFusión" && self.condicionParaMrFusion() method condicionParaObrero() = self.mayorDeEdad() && self.fuerza() > 50 method condicionParaMrFusion() = self.mayorDeEdad() && self.conocimiento() > estacionEspacial.conocimientoPromedio() * 1.21 method rolActual(otroRol){ if (self.podesCambiarA(otroRol)) { rolActual = otroRol // Se requiere que solamente MrFusión // registre conocimientos if (self.rolActual() == "Libre" self.rolActual() == "Obrero") conocimientos = 0 return 0 } else { return "Error: No se pudo cambiar rol, no cumple requisito!" } } } </pre> | | <pre> #Tripulante (v.i. rolActual, edad, fuerza, conocimiento) >> mayorDeEdad ^ self edad >= 18 >> prestigio self rolActual = 'Libre' ifTrue: [^0]. self rolActual = 'Obrero' ifTrue: [^50]. self rolActual = 'MrFusión' ifTrue: [^100]. ^ -1 >> podesCambiarA: otroRol ^ (otroRol = 'Libre' or: [otroRol = 'Obrero' & self condicionParaObrero]) or: [otroRol = 'MrFusión' & self condicionParaMrFusion]. >> condicionParaObrero ^ self mayorDeEdad & (self fuerza > 50) >> condicionParaMrFusion ^ self mayorDeEdad & (self conocimiento > (EstacionEspacial conocimientoPromedio * 1.21)) >> rolActual: otroRol (self podesCambiarA: otroRol) ifTrue: [rolActual := otroRol. "Se requiere que solamente MrFusión registre conocimientos" (self rolActual = 'Libre' self rolActual = 'Obrero') ifTrue: [conocimientos := 0]] ifFalse: [^ 'Error: No se pudo cambiar rol, no cumple requisito!'] </pre> |

Asumiendo que el objeto "estaciónEspacial" existe y está bien codificado, se pide:

- Responder verdadero o falso y justificar en todos los casos:
 - Estaría mal que Libre, Obrero y MrFusion sean subclases de Tripulante.
 - Puede agregarse un rol Capitán sin necesidad de modificar código existente (el rol Capitán es como Mr Fusión, pero además de ser inteligente debe ser fuerte, con fuerza mayor a 73).
 - Puede agregarse el rol Capitán sin necesidad de repetir código.
 - Suponiendo que en la estación hay varios tripulantes menores de edad. Puedo hacer:

```

Wollok: tripulantes.forEach({ tripu => tripu.rolActual("Obrero") })
Smalltalk: tripulantes do: [:tripu | tripu rolActual: 'Obrero' ]

```

Y jamás me enteraría de que hubo un error.
- Reescribir la solución (incluyendo al capitán) solucionando los problemas descubiertos y utilizando correctamente los conceptos vistos en la materia.
- Explicar conceptualmente los cambios realizados, indicando cómo se resuelven los problemas encontrados en el punto 1.