
Examen de Prácticas - 14 de enero de 2020
LTP (Tipo B)

ALUMNO: _____ GRUPO: _____

Instrucciones

- El alumno dispone de 60 minutos para resolver el examen.
- El examen consta de 5 preguntas que deberán responderse en el mismo enunciado, en los recuadros incluidos en cada pregunta.

Parte 1 – Haskell

Considera disponible la siguiente definición de la clase de tipos **Shape**:

```
class (Show a, Eq a) => Shape a where
    area :: a -> Float
    perimeter :: a -> Float
    distance :: a -> Float
```

Donde **area**, **perimeter** y **distance** son funciones tales que, dada una figura (dato de un tipo instancia de la clase **Shape**), calculan su área, perímetro y distancia al origen, respectivamente.

Pregunta 1.1 – Haskell (2.20 puntos)

Define una función **ins** cuyo tipo es:

```
ins :: (Shape a) => a -> [a] -> [a]
```

Dadas una figura **f** y una lista **lf** de figuras ordenadas ascendentemente por su distancia al origen, **ins f lf** devuelve una lista que contiene, ordenadas por sus distancias al origen, la figura **f** y las figuras de la lista **lf**.

Ejemplo de uso. Suponiendo disponible **Rectangle** como tipo de la clase **Shape**. Sea **r1** un rectángulo en el punto (2,7) de anchura 4 y altura 8, y **r2** un rectángulo en el origen de anchura 3 y altura 2, entonces:

```
*Main> let r1 = Rectangle (2.0,7.0) 4.0 8.0
*Main> let r2 = Rectangle (0.0,0.0) 3.0 2.0
*Main> ins (Rectangle (1.0,2.0) 9.0 7.0) [r2, r1]
[Rectangle (0.0,0.0) 3.0 2.0,Rectangle (1.0,2.0) 9.0 7.0,Rectangle (2.0,7.0) 4.0 8.0]
```

SOLUCIÓN RECURSIVA:

```
ins x [] = [x]
ins x (y:ys)
    | distance x <= distance y = x : y : ys
    | otherwise = y : ins x ys
```

SOLUCIÓN CON LISTAS INTENSIONALES:

```
ins x y = [z | z <- y, distance z < distance x]
        ++ [x] ++ [z | z <- y, distance z >= distance x]
```

Pregunta 1.2 – Haskell (2.20 puntos)

Define una función `perimeterSum` cuyo tipo es:

```
perimeterSum :: (Shape a) => [a] -> Float
```

La función `perimeterSum`, dada una lista de figuras, devuelve la suma de sus perímetros.

Ejemplo de uso. Suponiendo lo mismo que en el ejemplo de la pregunta 1.1:

```
*Main> perimeterSum [r1, r2]
34.0
```

SOLUCIÓN RECURSIVA:

```
perimeterSum [] = 0
perimeterSum (x:xs) = perimeter x + perimeterSum xs
```

SOLUCIÓN CON LISTAS INTENSIONALES:

```
perimeterSum x = sum [perimeter y | y <- x]
```

SOLUCIÓN CON FUNCIÓN MAP:

```
perimeterSum x = sum (map perimeter x)
```

Pregunta 1.3 – Haskell (2.20 puntos)

Define una función `areaFilter` cuyo tipo es:

```
areaFilter :: (Shape a) => [a] -> Float -> [a]
```

Dada una lista de figuras `lf` y un número real `d`, `areaFilter lf d` devuelve una lista con las figuras de `lf` que tienen un área mayor a `d`.

Ejemplo de uso. Suponiendo lo mismo que en el ejemplo de la pregunta 1.1:

```
*Main> areaFilter [r1, r2] 9.0
[Rectangle (2.0,7.0) 4.0 8.0]
```

SOLUCIÓN CON LISTAS INTENSIONALES:

```
areaFilter x y = [z | z <- x, (area z) > y]
```

SOLUCIÓN RECURSIVA:

```
areaFilter [] _ = []
areaFilter (x:xs) y
  | area x > y = x : areaFilter xs y
  | otherwise = areaFilter xs y
```

Parte 2 – Prolog

Dada la siguiente base de conocimiento (mostrada a 2 columnas):

<pre>% ibiza es un modelo de la marca seat model('ibiza', 'seat'). model('cordoba', 'seat'). model('altea', 'seat'). model('golf', 'volkswagen'). model('touran', 'volkswagen'). model('clio', 'renault'). model('twingo', 'renault'). model('megane', 'renault'). model('scenic', 'renault'). model('2008', 'peugeot'). model('3008', 'peugeot'). model('corsa', 'opel'). % seat es una marca fabricada en españa country('seat', 'españa'). country('renault', 'francia'). country('peugeot', 'francia'). country('volkswagen', 'alemania'). country('opel', 'alemania'). % ibiza es un modelo fabricado desde 1984 since('ibiza', 1984). since('cordoba', 1993).</pre>	<pre>since('altea', 2004). since('golf', 1974). since('touran', 2003). since('clio', 1990). since('twingo', 1993). since('megane', 1995). since('scenic', 1995). since('2008', 2013). since('3008', 2008). since('corsa', 1982). % ibiza es un modelo del Segmento B segment('ibiza', 'b'). segment('cordoba', 'b'). segment('altea', 'c'). segment('golf', 'c'). segment('touran', 'c'). segment('clio', 'b'). segment('twingo', 'a'). segment('megane', 'c'). segment('scenic', 'c'). segment('2008', 'b'). segment('3008', 'c'). segment('corsa', 'b').</pre>
---	---

Pregunta 2.1 – Prolog (1.70 puntos)

Define un predicado `countriesTwoBrands` que permita encontrar los países que tengan al menos dos marcas de coches. Ejemplo de uso: `?- countriesTwoBrands(C).`

```
C = francia ;
C = alemania.
```

SOLUCIÓN:

```
countriesTwoBrands(C) :- country(B1, C), country(B2, C), B1 @< B2.
```

Pregunta 2.2 – Prolog (1.70 puntos)

Define un predicado `oldiesModelsByBrand` que permita encontrar los modelos de una marca dada y anteriores a un año dado. Ejemplo de uso: `?- oldiesModelsByBrand(M, renault, 1995).`

```
M = clio ;
M = twingo.
```

SOLUCIÓN:

```
oldiesModelsByBrand(M, B, Y) :- model(M, B), since(M, A), A < Y.
```