
Examen de Prácticas - 14 de enero de 2019
LTP (Tipo A)

ALUMNO: _____ GRUPO: _____

Instrucciones

- El alumno dispone de 60 minutos para resolver el examen.
- El examen consta de 3 preguntas que deberán responderse en el mismo enunciado, en los recuadros incluidos en cada pregunta.

Pregunta 1 – Haskell (3.25 puntos)

Considera disponible una función `member` cuyo tipo es:

```
member :: (Eq a) => a -> [a] -> Bool
```

La función `member` comprueba si un elemento está en una lista de elementos (del mismo tipo `a`, genérico, pero restringido a la clase de tipos `Eq`).

Define una función `intersect` cuyo tipo sea:

```
intersect :: (Eq a) => [a] -> [a] -> [a]
```

La función `intersect`, dadas dos listas de elementos del tipo `(Eq a) => a`, devuelve una lista que es la intersección de las dos listas recibidas. Considera listas sin elementos repetidos.

Ejemplos de uso:

```
*Main> intersect [3,4,2,1,5] [6,5,2,8,1]
[2, 1, 5]
*Main> intersect [3,4,2] [6,5,2,8,1]
[2]
*Main> intersect [3,4,2] [6,5,8,1]
[]
*Main> intersect ['a','b','k','h'] ['h','n','b']
"bh"
*Main> intersect [2.0, 3.1] [4.5, 3.1, 2.2]
[3.1]
```

El orden de los elementos en la lista resultado no es relevante. Así, en los ejemplos, sería un resultado válido cualquier lista que contuviera los mismos valores, pero en otro orden.

SOLUCIÓN CON RECURSIÓN DIRECTA:

```
intersect :: (Eq a) => [a] -> [a] -> [a]
intersect [] _ = []
intersect (x:xs) y
  | member x y = x : intersect xs y
  | otherwise = intersect xs y
```

SOLUCIÓN CON LISTAS INTENSIONALES:

```
intersect :: (Eq a) => [a] -> [a] -> [a]
intersect x y = [z | z <- x, member z y]
```

Pregunta 2 – Haskell (3.25 puntos)

Considera la definición del tipo de dato `BinTreeInt`:

```
data BinTreeInt = Void | Node Int BinTreeInt BinTreeInt deriving Show
```

Define una función `mapTree` cuyo tipo sea:

```
mapTree :: (Int -> Int) -> BinTreeInt -> BinTreeInt
```

La función `mapTree`, dada una función de entero a entero y un árbol binario de enteros, devuelve un nuevo árbol binario de enteros, resultado de aplicar la función a todos los enteros almacenados en los nodos del árbol pasado como segundo argumento.

Ejemplos de uso:

```
*Main> mapTree (+3) Void
Void
*Main> mapTree (+3) (Node 4 Void Void)
Node 7 Void Void
*Main> mapTree (+3) (Node 3 (Node 2 Void Void) (Node 7 (Node 4 Void Void) (Node 9 Void Void)))
Node 6 (Node 5 Void Void) (Node 10 (Node 7 Void Void) (Node 12 Void Void))
*Main> mapTree (*3) (Node 3 (Node 2 Void Void) (Node 7 (Node 4 Void Void) (Node 9 Void Void)))
Node 9 (Node 6 Void Void) (Node 21 (Node 12 Void Void) (Node 27 Void Void))
```

SOLUCIÓN CON RECURSIÓN DIRECTA:

```
mapTree :: (Int -> Int) -> BinTreeInt -> BinTreeInt
mapTree _ Void = Void
mapTree f (Node x izq der) = Node (f x) (mapTree f izq) (mapTree f der)
```

Pregunta 3 – Prolog (3.50 puntos)

Resolver los 2 ejercicios que se plantean, dada la siguiente base de conocimiento:

```
/* movie(M, Y), M is a movie released in the year Y */
movie(barton_fink, 1991).
movie(the_big_lebowski, 1998).
movie(fargo, 1996).
movie(lick_the_star, 1998).
movie(mission_impossible, 1996).
movie(fall, 1997).
/* director(M, D), M is a movie directed by D */
director(the_big_lebowski, joel_coen).
director(barton_fink, ethan_coen).
director(barton_fink, joel_coen).
director(fargo, ethan_coen).
director(fargo, joel_coen).
director(lick_the_star, sofia_coppola).
director(mission_impossible, brian_de_palma).
/* actor(M, A, R), the actor A played the role of R in the movie M */
actor(mission_impossible, tom_cruise, ethan_hunt).
actor(mission_impossible, jon_voight, jim_phelps).
actor(barton_fink, john_turturro, barton_fink).
actor(barton_fink, john_goodman, charlie_meadows).
actor(the_big_lebowski, jeff_bridges, jeffrey_lebowski__the_dude).
actor(the_big_lebowski, john_goodman, walter_sobchak).
```

1.- Define un predicado moviesTwoDirectors que permita encontrar las películas que tengan al menos dos directores. Ejemplo de uso:

```
?- moviesTwoDirectors(M).
    M = barton_fink ;
    M = fargo.
```

SOLUCIÓN:

```
moviesTwoDirectors(M) :- director(M, D1), director(M, D2), D1 @< D2.
```

2.- Define un predicado moviesAfter que permita encontrar las películas estrenadas después de un año dado. Ejemplo de uso:

```
?- moviesAfter(M, 1997).
    M = the_big_lebowski ;
    M = lick_the_star.
```

SOLUCIÓN:

```
moviesAfter(M, A) :- movie(M, Y), Y > A.
```