Examen de Prácticas - 25 de enero de 2021
LTP (Tipo A)

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.

## Pregunta 1 – Haskell (2.20 puntos)

Define una función `separate` cuyo tipo es: `separate :: Int -> [Int] -> [Int]`

Dados `e`, un entero, y `le`, una lista de enteros, `(separate e le)` devuelve una lista con los elementos de `le` reubicados del siguiente modo: los elementos menores o iguales a `e` se sitúan antes que los elementos mayores que `e`.
Por ejemplo, el resultado de `separate 0 [3,-1,0,2,-4]` contendría los valores `[-1,0,-4]` seguidos de los valores `[3,2]`, sin importar el orden relativo dentro de estos dos grupos de valores (serían válidos los siguientes resultados: `[-1,0,-4,3,2]`, o `[-1,-4,0,2,3]`, o `[0,-1,-4,3,2]` , o ...).

**REQUISITO**: Se debe resolver mediante recursión o mediante listas intensionales.

**Ejemplo de uso.** Se muestran, para los mismos datos de entrada a la función, dos resultados distintos, ambos válidos por cumplir lo indicado anteriormente.

```
*Main> let lis = [3,7,2,1,3,8,4,5,7,9,0]
*Main> separate 3 lis
[3,2,1,3,0,7,8,4,5,7,9]
```

```
*Main> let lis = [3,7,2,1,3,8,4,5,7,9,0]
*Main> separate 3 lis
[3,2,1,3,0,9,7,5,4,8,7]
```

Solución con listas intensionales:

```
separate :: Int -> [Int] -> [Int]
separate x y = [z | z <- y, z <= x] ++ [z | z <- y, z > x]
```

Solución recursiva:

```
separate :: Int -> [Int] -> [Int]
separate _ [] = []
separate x (y:ys)
   | y <= x = y : separate x ys
   | otherwise = separate x ys ++ [y]
```

## Pregunta 2 – Haskell (2.20 puntos)

Considera disponible la siguiente definición, correspondiente a árboles binarios, ordenados, de enteros:

```
data BinTreeInt = Void | Node Int BinTreeInt BinTreeInt deriving (Eq, Show)
```

Define una función `getMinPath` cuyo tipo es: `getMinPath :: BinTreeInt -> [Int]`

Dado un árbol de tipo `BinTreeInt`, la función devuelve una lista con los enteros que hay en el camino desde la raíz del árbol hasta su hoja más a la izquierda (el valor mínimo del árbol).

**Ejemplo de uso.**

```
*Main> let tree = (Node 5 (Node 3 (Node 1 Void Void)(Node 4 Void Void)) (Node 6 Void
(Node 8 Void Void)))
*Main> getMinPath tree
[5,3,1]
```

Solución:

```
getMinPath :: BinTreeInt -> [Int]
getMinPath Void = []
getMinPath (Node x izq _) = x : getMinPath izq
```

## Pregunta 3 – Haskell (2.20 puntos)

Considera disponibles las siguientes definiciones:

```
type Title = String
type Authors = [String]
type Pages = Int
type Chapters = Int
data Obra = Obra Title Authors
data Textual = Textual Obra Pages Chapters
class Structure a where
   items :: a -> Int
   isShort :: a -> Bool
```

Donde:

- `Obra` es un tipo de datos que representa obras artísticas con los siguientes valores:
  - `Title`, un string que es el título de la obra.
  - `Authors`, una lista de string que contiene los nombres de los autores de la obra.
- `Textual` es un tipo de datos que representa obras artísticas de texto, con los siguientes valores:
  - `Obra`, con el título y los autores de la obra.
  - `Pages`, un entero igual al número de páginas.
  - `Chapters`, un entero igual al número de capítulos.
- `Structure` es una clase de tipos que define 2 funciones para los tipos `a` que la instancien:

- `items` que devuelve el número de partes o elementos estructurales del dato de tipo `a`.
- `isShort` que devuelve un valor lógico indicando si el dato de tipo `a` es corto, estructuralmente.

**Se pide:** Instanciar la clase `Structure` para el tipo `Textual` teniendo en cuenta que:

- `items` ha de devolver el número de capítulos del dato de tipo `Textual`.
- `isShort` ha de devolver `True` si el número de páginas es menor que 100 o si el número de capítulos es menor que 5, y ha de devolver `False` en caso contrario.

**Ejemplo de uso.**

```
*Main> let authors = ["G. Brassard", "P. Bratley"]
*Main> let book = Textual (Obra "Fundamentos de Algoritmia" authors) 579 13
*Main> items book
13
*Main> isShort book
False
```

Solución:

```
instance Structure Textual where
   items (Textual _ _ c) = c
   isShort (Textual _ p c) = p < 100 || c < 5
```

## Pregunta 4 – Prolog (1.70 puntos)

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

```
% libro "Ana Karenina", autor "L. Tolstoi"
libro("Ana Karenina", "L. Tolstoi").
libro("La broma", "M. Kundera").
libro("El castillo", "F. Kafka").
libro("Tiempos recios", "M. Vargas Llosa").
libro("Niebla", "M. Unamuno").
libro("Hamlet", "W. Shakespeare").
libro("Odessa", "F. Forsyth").
libro("El pirata", "J. Conrad").
libro("Jane Eyre", "C. Bronte").
libro("El unicornio", "M. Mujica Lainez").
```

```
% "Ana" lee "Ana Karenina" desde 29-dic-20
prestamo("Ana", "Ana Karenina", date(29,dic,20)).
prestamo("Juan", "La broma", date(29,dic,20)).
prestamo("Pepe", "El castillo", date(29,dic,20)).
prestamo("Paco", "Tiempos recios", date(29,dic,20)).
prestamo("Pepe", "Niebla", date(31,dic,20)).
prestamo("Juan", "Hamlet", date(7,ene,21)).
prestamo("Alicia", "Odessa", date(7,ene,21)).
prestamo("Pepe", "El pirata", date(21,ene,21)).
prestamo("Ana", "Jane Eyre", date(21,ene,21)).
prestamo("Ana", "El unicornio", date(21,ene,21)).
prestamo("Juan", "Niebla", date(21,ene,21)).
```

Define un predicado `whoReadsWho` que permita consultar las personas que leen libros de un autor dado, o bien los autores cuyos libros lee una persona dada, en ambos casos en un mes y año dados.

**Ejemplos de uso.**

```
?- whoReadsWho(X,"M. Unamuno",M,Y).
X = "Pepe",
M = dic,
Y = 20 ;
X = "Juan",
M = ene,
Y = 21.
```

```
?- whoReadsWho("Ana",X,_,Y).
X = "L. Tolstoi",
Y = 20 ;
X = "C. Bronte",
Y = 21 ;
X = "M. Mujica Lainez",
Y = 21.
```

Solución:

```
whoReadsWho(P,A,M,Y) :- prestamo(P,B,date(_,M,Y)), libro(B,A).
```

## Pregunta 5 – Prolog (1.70 puntos)

Dada la siguiente base de conocimiento:

```
% "Fundamentos de Algoritmia", de "G. Brassard" y "P. Bratley",
% libro editado por "Prentice Hall" en 1997
book("Fundamentos de Algoritmia", ["G. Brassard", "P. Bratley"], "Prentice Hall", 1997).
book("Sistemas Operativos", ["William Stallings"], "Prentice Hall", 1997).
book("Fundamentos de Bases de Datos", ["H. Korth", "A. Silberschatz"], "McGrawHill", 1993).
book("Fisica Cuantica", ["Robert Eisberg", "Robert Resnick"], "Limusa", 1979).
book("Sistemas Operativos", ["Milan Milenkovic"], "McGrawHill", 1994).
```

Define un predicado `authorsOf` que permita consultar los autores de libros de un título dado o bien los títulos de libros escritos (solo o en colaboración) por un autor dado.
Si se considera necesario, pueden usarse predicados predefinidos como, por ejemplo, `member` o `append`.

**Ejemplos de uso.**

```
?- authorsOf("Sistemas Operativos",X).
X = "William Stallings" ;
X = "Milan Milenkovic".
```

```
?- authorsOf(B,"Robert Eisberg").
B = "Fisica Cuantica" .
```

Solución:

```
authorsOf(B, X) :- book(B, L, _, _), member(X, L).
```