Programación Funcional - Enunciado de Práctica 1 - 2021/22

La práctica consiste escribir un programa en Haskell para crear el calendario de cualquier año e imprimirlo en pantalla en 3 ó 4 columnas.

Ejemplo Para obtener el calendario en 3 columnas del año 2021 evaluaremos la expresión:

```
*Calendarios> printCalendario 3 2021
```

```
Enero 2021                  Febrero 2021                Marzo 2021

Lu Ma Mi Ju Vi Sa Do        Lu Ma Mi Ju Vi Sa Do        Lu Ma Mi Ju Vi Sa Do
             1  2  3         1  2  3  4  5  6  7         1  2  3  4  5  6  7
 4  5  6  7  8  9 10         8  9 10 11 12 13 14         8  9 10 11 12 13 14
11 12 13 14 15 16 17        15 16 17 18 19 20 21        15 16 17 18 19 20 21
18 19 20 21 22 23 24        22 23 24 25 26 27 28        22 23 24 25 26 27 28
25 26 27 28 29 30 31                                    29 30 31


Abril 2021                  Mayo 2021                   Junio 2021

Lu Ma Mi Ju Vi Sa Do        Lu Ma Mi Ju Vi Sa Do        Lu Ma Mi Ju Vi Sa Do
          1  2  3  4                        1  2           1  2  3  4  5  6
 5  6  7  8  9 10 11         3  4  5  6  7  8  9         7  8  9 10 11 12 13
12 13 14 15 16 17 18        10 11 12 13 14 15 16        14 15 16 17 18 19 20
19 20 21 22 23 24 25        17 18 19 20 21 22 23        21 22 23 24 25 26 27
26 27 28 29 30              24 25 26 27 28 29 30        28 29 30
                            31


Julio 2021                  Agosto 2021                 Septiembre 2021

Lu Ma Mi Ju Vi Sa Do        Lu Ma Mi Ju Vi Sa Do        Lu Ma Mi Ju Vi Sa Do
          1  2  3  4                           1              1  2  3  4  5
 5  6  7  8  9 10 11         2  3  4  5  6  7  8         6  7  8  9 10 11 12
12 13 14 15 16 17 18         9 10 11 12 13 14 15        13 14 15 16 17 18 19
19 20 21 22 23 24 25        16 17 18 19 20 21 22        20 21 22 23 24 25 26
26 27 28 29 30 31           23 24 25 26 27 28 29        27 28 29 30
                            30 31


Octubre 2021                Noviembre 2021              Diciembre 2021

Lu Ma Mi Ju Vi Sa Do        Lu Ma Mi Ju Vi Sa Do        Lu Ma Mi Ju Vi Sa Do
             1  2  3         1  2  3  4  5  6  7              1  2  3  4  5
 4  5  6  7  8  9 10         8  9 10 11 12 13 14         6  7  8  9 10 11 12
11 12 13 14 15 16 17        15 16 17 18 19 20 21        13 14 15 16 17 18 19
18 19 20 21 22 23 24        22 23 24 25 26 27 28        20 21 22 23 24 25 26
25 26 27 28 29 30 31        29 30                       27 28 29 30 31
```

Para la práctica, disponemos de un **módulo Calendarios** donde se proporcionan algunas funciones (se dan ya definidas o bien se da su especificación informal), de forma que dirijan una manera de abordar el problema a resolver.

El resto de funciones necesarias deben ser declaradas y definidas de forma clara y precisa, haciendo uso de los esquemas recursivos estudiados y demás características propias del paradigma funcional.

```
module Calendarios where

type Dibujo = [Linea]  -- cada dibujo es una lista de líneas (de igual longitud)
type Linea = [Char]    -- cada línea es una lista de caracteres
type Year = Int
type Columna = Int       -- será 3 ó 4

-- Para imprimir un dibujo en pantalla:
printDibujo :: Dibujo -> IO()
printDibujo dib = do
                  putStr "\n"
                  (putStr . concat . map (++"\n")) dib

-- Para imprimir el calendario de un año con un número de columnas:
printCalendario :: Columna ->  Year -> IO()
printCalendario c a = printDibujo (calendario c a)
```

Función principal: **calendario**

```
calendario :: Columna -> Year -> Dibujo
-- el dibujo de un calendario en c columnas de un año dado
calendario c  =  bloque c . map dibujomes . meses
```

donde:

1) La función **meses** devuelve, para un año dado, la información relevante de cada mes.
Para el primer día del mes usaremos: 1=lunes,…, 7=domingo.

```
meses ::  Year -> [(String, Year, Int, Int)]
-- meses n devuelve una lista de 12 elementos con los datos relevantes de cada uno de
-- los meses del año n: (nombre del mes, n, primer día del mes, longitud del mes)
```

Ejemplo:

```
*Calendarios> meses 2021
[("Enero",2021,5,31),("Febrero",2021,1,28),("Marzo",2021,°,31),("Abril",
2021,4,30),("Mayo",2021,6,31),("Junio",2021,2,30),("Julio",2021,4,31),("A
gosto",2021,7,31),("Septiembre",2021,3,30),("Octubre",2021,5,31),("Noviem
bre",2021,1,30),("Diciembre",2021,3,31)]
```

2) La función **dibujomes**, dada la información relevante del mes, devuelve su dibujo.

```
dibujomes ::(String, Year, Int, Int) -> Dibujo
-- dibujomes (nm,a,pd,lm) devuelve un dibujo de dimensiones 10x25 formado
-- por el titulo y la tabla del mes de nombre nm y año a.
-- Necesita como parámetros el primer dia pd y la longitud del mes lm
```

Ejemplo:

```
*Calendarios> dibujomes ("Octubre",2021,4,31)
[" Octubre 2021            ","                          ","  Lu Ma Mi Ju Vi
Sa Do    ","     1  2  3  4  5  6    ","  7  8  9 10 11 12 13    ","  14
15 16 17 18 19 20    ","  21 22 23 24 25 26 27    ","  28 29 30 31
","                          ","                          "]
```

de forma que si le aplicamos la función printDibujo quedará el dibujo del mes (dimensiones 10x25):

```
*Calendarios> printDibujo (dibujomes ("Octubre",2021,5,31))
```

```
 Octubre 2021

 Lu Ma Mi Ju Vi Sa Do
              1  2  3
  4  5  6  7  8  9 10
 11 12 13 14 15 16 17
 18 19 20 21 22 23 24
 25 26 27 28 29 30 31
```

```
*Calendarios> alto (dibujomes ("Octubre",2021,5,31))
10
*Calendarios> ancho (dibujomes ("Octubre",2021,5,31))
25
```

3) La función **bloque** nos permitirá agrupar los dibujos de una lista para hacer un solo dibujo

```
bloque :: Int -> [Dibujo] -> Dibujo
-- bloque n dibs es el dibujo formado al agrupar de n en n los dibujos de dibs,
-- extender cada sublista y luego apilarlas
```

4) Entre el resto de funciones, se os proporciona **ene1** y se debe diseñar la función **pdias** que servirá para calcular los primeros días de la semana de cada mes de un año:

```
ene1 :: Year -> Int
ene1 a = mod (a + div (a-1) 4 - div (a-1) 100 + div (a-1) 400) 7
-- ene1 a devuelve el dia de la semana del 1 de enero del año a
--        siendo 1=lunes, 2=martes, ..., 6=sabado, 0=domingo

pdias :: Year -> [Int]
-- pdias a    devuelve una lista con 12 dias que son los días de la semana
--            en que comienza cada mes del año a
--            siendo 1 = lunes, 2 = martes, ..., 6= sabado, 7=domingo
```

Ejemplos:

```
*Calendarios> pdias 2021
[5,1,1,4,6,2,4,7,3,5,1,3]
*Calendarios> pdias 2020       -- 2020 es bisiesto
[3,6,7,3,5,1,3,6,2,4,7,2]
*Calendarios> pdias 2022
[6,2,2,5,7,3,5,1,4,6,2,4]
```