Contents

Chapter 1

Labo1

```
module Labo1 (
    time, yearsSeg, daysSeg, hoursSeg, minsSeg, years, days, hours,
    mins, segs, sol1a, sol1b, sol2A, sol2B, sol3, calculoDigitos,
    sumaDigitos, reduccion, bools
  ) where
```

1.1 Ejercicio 1

1.1.1 A

`time :: Integer`

tiempo base del ejercicio 1

`yearsSeg :: Integer`

Segundos totales en un año

`daysSeg :: Integer`

Segundos totales en un día

`hoursSeg :: Integer`

Segundos totales en una hora

`minsSeg :: Integer`

Segundos totales en un minuto

`years :: Integer`

Caclulo de los años equivalentes a `time`

`days :: Integer`

Calculo de las horas equivalentes a `time` quitando `years`

`hours :: Integer`

Calculo de las horas equivalentes a `time` quitando `days` y `years`

`mins :: Integer`

Calculo de las horas equivalentes a `time` quitando `days`, `years` y `hours`

`segs :: Integer`

Calculo de las horas equivalentes a `time` quitando `days`, `years`, `hours` y `mins`

`sol1a :: (Integer, Integer, Integer, Integer, Integer)`

Tupla con el tiempo total de `time` en años, dias, horas, minutos y segundos

1.1.2 B

```
sol1b :: Integer -> (Integer, Integer, Integer, Integer, Integer)
```

Descripción

Coje el tiempo dado y lo convierte a (Años, Días, Horas, Minutos, Segundos)

Ejemplos

```
>>> sol1b 1000000
(0,11,13,46,40)
```

1.2 Ejercicio 2

1.2.1 A

```
sol2A ::  Integer -> Bool
```

Descripción

Coje el año dado y devuelve True si es bisiesto, False en caso contrario

Ejemplos

```
>>> sol2A 2008
True
```

```
>>> sol2A 2022
False
```

1.2.2 B

```
sol2B ::  Integer -> Bool
```

Descripción

Coje el año dado y devuelve True si es bisiesto, False en caso contrario

Ejemplos

```
>>> sol2A 2008
True


>>> sol2A 2022
False
```

1.3 Ejercicio 3

```
sol3 ::  (Fractional a1, Integral a2, Foldable t) => t a2 -> a1
```

Descripción

Error de tipado ya que la division / no trabaja con Integer

1.4 Ejercicio 4

```
calculoDigitos ::  Integer -> Integer
```

Descripción

Cálculo del número de dijitos de un numero dado

Ejemplos

```
>>> calculoDigitos 0
1


>>> calculoDigitos 1984
4
```

```
sumaDigitos ::  Integral t => t -> t
```

Descripción

Suma todos los digitos de un numero dado

Ejemplos

```
>>> sumaDigitos 0
0


>>> sumaDigitos 1984
22
```

```
reduccion ::  Integral t => t -> t
```

Descripción

Suma todos los digitos de un numero dado hasta que el total es menor que 10

Ejemplos

```
>>> reduccion 0
0


>>> reduccion 1984
4
```

1.5 Ejercicio 5

```
bools ::  Bool -> Bool -> Bool
```

Disyunción booleana definida por ajuste de patrones

Chapter 2

Labo2

```
module Labo2 (
    cuadrados, cuadradoInverso, rsumcos, sumMenores,
    siguientePrimo, factores, isPrime, iguales, menor, mayorA, ex,
    filter2, filters, mapx
  ) where
```

```
cuadrados ::  (Num b, Enum b) => b -> [b]

cuadradoInverso ::  (Num a, Enum a) => a -> [(a, a)]

rsumcos ::  (Floating a, Enum a) => a -> a

sumMenores ::  Integral a => a -> a

siguientePrimo ::  Integral a => a -> a

factores ::  Integral a => a -> [a]

isPrime ::  Integral a => a -> Bool

iguales ::  Eq b => (a -> b) -> (a -> b) -> a -> a -> Bool

menor ::  (Enum a, Num a) => a -> (a -> Bool) -> a

mayorA ::  Enum a => a -> a -> (a -> Bool) -> a

ex ::  Enum a => a -> a -> (a -> Bool) -> Bool
```

```
filter2 ::  [a] -> (a -> Bool) -> (a -> Bool) -> ([a], [a])

filters ::  [a] -> [a -> Bool] -> [[a]]

mapx ::  t -> [t -> b] -> [b]
```

Chapter 3

Labo3

```
module Labo3 (
    last', reverse', all', min', map', filter', takeWhile',
    listaNegPos, listaParejas', listaParejas, sufijos, sublistas,
    permuta, intercalado, sumandos, sumandos'
  ) where
```

```
last' ::  [a] -> a

reverse' ::  [a] -> [a]

all' ::  (a -> Bool) -> [a] -> Bool

min' ::  Ord a => [a] -> a

map' ::  (a -> b) -> [a] -> [b]

filter' ::  (a -> Bool) -> [a] -> [a]

takeWhile' ::  (a -> Bool) -> [a] -> [a]

listaNegPos ::  [Integer]

listaParejas' ::  (Num b, Enum b, Eq b) => b -> [(b, b)]

listaParejas ::  Integral a => [(a, a)]

sufijos ::  [a] -> [[a]]
```

```
sublistas ::  [a] -> [[a]]

permuta ::  [a] -> [[a]]

intercalado ::  a -> [a] -> [[a]]

sumandos ::  Integral a => a -> [[a]]

sumandos' ::  Integral a => a -> a -> a -> [[a]]
```

Chapter 4

Labo4

```
module Labo4 (
    Punto(P), pointSum,
    Direccion(DERECHA, IZQUIERDA, ABAJO, ARRIBA), mueve, destino,
    trayectoria, Nat(Suc, Cero), (~+), (~*), natToInt, Complejo(C),
    Medible(medida)
  ) where
```

```
data Punto
```

Constructors

```
=  P Int Int
```

```
instance Show Punto
```

```
pointSum ::  (Ord a, Num a) => a -> a -> a
```

```
data Direccion
```

Constructors

```
=  ARRIBA
|  ABAJO
|  IZQUIERDA
|  DERECHA
```

```
instance Eq Direccion
instance Ord Direccion
instance Show Direccion
```

```
mueve ::  Punto -> Direccion -> Punto
```

Moves a point in a given direction

```
destino ::  Punto -> [Direccion] -> Punto
```

moves a point across a list of directions

```
trayectoria ::  Punto -> [Direccion] -> [Punto]
```

Shows a list of points where a point has been moving given a direcction list

```
data Nat
```

Constructors

```
=   Cero
|   Suc Nat
```

```
instance Eq Nat
instance Ord Nat
instance Show Nat
```

```
(~+) ::  Nat -> Nat -> Nat

(~*) ::  Nat -> Nat -> Nat

natToInt ::  Nat -> Int
```

```
data Complejo
```

Constructors

```
=   C Float Float
```

```
instance Eq Complejo
instance Fractional Complejo
instance Num Complejo
instance Show Complejo
```

```
class Medible a where
```

Methods

```
     medida ::  a -> Int


instance Medible Bool
instance Medible a => Medible [a]
instance (Medible a, Medible b) => Medible (a, b)
```

Chapter 5

Labo5

```
module Labo5 (
    getInt, adivina, Matriz, getFloat, getMatriz, getFilas,
    getDatosMatriz, dibujaMatriz, mostrarMatrizNueva, formatea,
    formatear, justify, addSpaces
  ) where
```

```
getInt ::  IO Int

adivina ::  Int -> IO ()


type Matriz = [[Float]]


getFloat ::  IO Float

getMatriz ::  IO Matriz

getFilas ::  Int -> IO [Float]

getDatosMatriz ::  Int -> Int -> IO Matriz

dibujaMatriz ::  Matriz -> IO ()

mostrarMatrizNueva ::  IO ()

formatea ::  String -> String -> Int -> IO ()
```

```
formatear ::  Int -> String -> String

justify ::  Int -> String -> String

addSpaces ::  Int -> String -> String
```