MODELOS AVANZADOS DE COMPUTACIÓN

PRÁCTICA 2: PRIMEROS PASOS CON FUNCIONES EN HASKELL

EJERCICIO 1: Sea la función **nsobrek** tal que **nsobrek n k** es el número de combinaciones de n elementos tomados de k en k; es decir:

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

Se pide: definir la función en **haskell** en el mayor número de variantes que hemos visto en clase.

EJERCICIO 2: Definir la función raíces tal que raíces a b c es la lista de las raices de la ecuación $ax^2 + bc + c = 0$.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Se pide: definir la función en **haskell** en el mayor número de variantes que hemos visto en clase.

EJERCICIO 3: Sucesión de Fibonacci

Los intervalos cerrados se pueden representar mediante -- una lista de dos números (el primero es el extremo inferior del intervalo y el segundo el superior).

Definir la función **interseccion :: Ord a => [a] -> [a] -> [a],** tal que (interseccion i1 i2) es la intersección de los intervalos i1 e-- i2. Por ejemplo,

interseccion [] [3,5] == []

interseccion [3,5] [] == []

interseccion [2,4] [6,9] == []

interseccion [2,6] [6,9] == [6,6]

interseccion [2,6] [0,9] == [2,6]

interseccion [4,6] [0,4] == [4,4]

interseccion [5,6] [0,4] == []

Se pide: definir la función en **haskell** en el mayor número de variantes que hemos visto en clase.

EJERCICIO 4: Comprobar la pertenencia a una lista usando una función recursiva.

Pertenece a [b]

Ejemplos:

Pertenece 3 [2,3,5] == True
Pertenece 4 [2,3,5] == False

Se pide: definir la función en **haskell** en el mayor número de variantes que hemos visto en clase.

Redefinir la función para utilizar tupas y listas de tuplas

Ejemplos:

Pertenece (5,4) [(5,4),(4,7)] == True

Pertenece (5,4) [(3,4),(4,7)] == False

EJERCICIO 5: Seleccionar 2 retos del proyecto de Euler que se puedan resolver con *lo visto hasta la sesión del día 24 de Octubre,* e implementarlo con el mayor número de definiciones posibles.

https://projecteuler.net/

ENTREGA y MEMORIA

La entrega se realizará a través de la plataforma en un documento .ZIP, el cual debe contener:

- Un fichero apellido1-apellido2-nombre_P2***.hs*** con las diferentes implementaciones por cada ejercicio
- Un documento APELLIDO1-APELLIDO2-NOMBRE_P2_Memoria.pdf que deberá contener lo siguiente:
  - Portada
  - Índice
  - Código fuente de cada implementación
  - Descripción de cada código.
- Los códigos fuentes en la memoria deben ir presentados mediante una imagen generada https://carbon.now.sh/ con las configuración propuesta en el fichero carbon-config.json que se proporciona en la plataforma. La apariencia debe quedar de la siguiente manera:

```haskell
{- FACTORIAL GUARDAS -}

--factorial_guarda ::Integer->Integer
--factorial_guarda :: (Num a, Ord a) => a -> a
factorial_guarda :: (Num a, Ord a) => a -> a
factorial_guarda n
    | n ==0     = 1
    | n > 0     = n * factorial_guarda(n-1)
    | otherwise = error "valor negativo"

```

FECHA LÍMITE DE ENTREGA

Miércoles 31 de Octubre a las 18:00