Práctica 2: Funciones en haskell

Alba Márquez Rodríguez

Ingeniería Informática, Universidad de Huelva

Modelos Avanzados de Computación

## Ejercicio 1: Nsobrek

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

**Texto

Descripción generada automáticamente con confianza media**

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

Guardas:

Texto

Descripción generada automáticamente

If-then else:

Texto

Descripción generada automáticamente

Case:

Texto

Descripción generada automáticamente

Primero ya que debemos calcular el factorial de varios números realizamos esta función para llamarla en la función principal.

En la función principal aseguramos que n > k ya que en el caso contrario el factorial de n-k será erróneo. Una vez se haya tenido esto en cuenta se calculan los factoriales y se operan tal y como se muestra en la fórmula de nsobrek.

Texto

Descripción generada automáticamente

## Actividad 2

**Definir la función raíces tal que raíces a b c es la lista de las raíces de la ecuación ax2 + bc + c = 0.**

Imagen que contiene Tabla

Descripción generada automáticamente

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

Guardas:

**Texto

Descripción generada automáticamente**

If:

**Texto

Descripción generada automáticamente**

Para calcular la raíz cuadrada primero se verifica el cálculo más básico, q es el caso en el que a = 0. Luego en el caso en el que pueda salir un valor no válido como que sea del tipo 0x^2 + 0x + c = c. Es decir, ninguna ecuación que resolver.

O bien, en el caso en el que la raíz fuese negativa y entonces no se podría calcular.

Tras verificar estas condiciones se procede al cálculo de la raíz, que se devuelve en una lista que ya la raíz puede tener valor positivo o negativo y hay que hacer dos cálculos.

## Actividad 3: Sucesión de Fibonacci

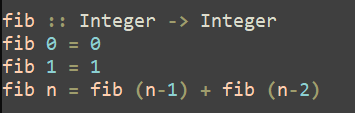
**Los números de Fibonacci quedan definidos por las ecuaciones**

Tabla

Descripción generada automáticamente

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

Local:



Guardas:

Texto

Descripción generada automáticamente con confianza media

If-then else:

Texto

Descripción generada automáticamente

Se realiza recursividad, esta terminará cuando n sea igual a 0 o 1. Así que estas son las primeras condiciones, en el caso en el que n no sea ni 0 o 1 habrá que seguir realizando esta recursividad sobre n-1 y n-2.

## Actividad 4: Comprobar la pertenencia a una lista usando una función recursive

**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.**

Guardas:

Texto

Descripción generada automáticamente

If-else-then:

Texto

Descripción generada automáticamente

Se crea una recursividad que se terminará en el caso en el que el elemento n se encuentre en la cabeza de la lista o la lista quede vacía.

Si no se cumple ninguna de estas condiciones hará recursividad sobre la cola de la lista (todos los elementos menos el primero, que es con el que se habrá comparado para saber si terminar la recursividad o continuarla).