PROGRAMACIÓN DECLARATIVA – HASKELL **FEBRERO 2002**

NOMBRE: **GRUPO:** **ESPECIALIDAD:**

1) ¿Cuál de las siguientes expresiones no es una lista válida en Haskell? (*0.5 puntos*)

a) [[ ]]
b) [1, [2,3]]
c) [(1,2), (3,4)]
d) [[1,2], [3,4]]
e) [(+), (-), (*)]

RESPUESTA: [ ]

2) ¿Cuál de las siguientes expresiones es equivalente a *filter p (map f xs)*? (*0.5 puntos*)

a) map f (filter p xs)
b) [*p (f x) | x <- xs*]
c) [*f x | x <- xs, p (f x)*]
d) *f* [*x | x <- xs, p x*]
e) [*f x | x <- xs, p x*]

RESPUESTA: [ ]

3) ¿Cuáles son los tipos de las siguientes funciones? (*0.75 puntos*)

uncurry f (x, y) = f x y *uncurry* :: _____

h x y = x == (x + y) *h* :: _____

curry f x y = f (x,y) *curry* :: _____

4) Escribe una función (dando su tipo) que inserte una elemento en su posición adecuada dentro de una lista ordenada. Por ejemplo (*0.75 puntos*)

inserta 5 [1,2,3,7,9] ==> [1,2,3,5,7,9]

5) Escribe una función *signos* (dando su tipo) que dada una lista de enteros devuelva una terna que indique cuántos son positivos, cuantos negativos y cuántos nulos. Por ejemplo (*1 punto*)

signos [1,-3,2,0,7,1] ==> (4,1,1)

6) Escribe, usando funciones de orden superior, una función *pega* (dando su tipo) que tome un elemento y una lista de listas y coloque dicho elemento en la cabeza de cada una de las listas. Por ejemplo (*1 punto*)

pega 10 [[1,2,3], [4,5], [9], [ ]] ==> [[10,1,2,3], [10,4,5], [10,9], [10]]

7) Dado el siguiente tipo para representar árboles binarios

data *Árbol a = Vacío | Nodo (Árbol a) a (Árbol a)* ***deriving*** *Show*

Dibuja el siguiente árbol: (*0.25 puntos*)

a1 = Nodo Vacío [1,2] (Nodo (Nodo Vacío [3,4,5] Vacío) [6] (Nodo Vacío [10] Vacío))

define una función *suma :: Árbol [Int] -> Int* que sume todos los elementos que aparecen en un árbol. Por ejemplo: (*0.75 puntos*)

suma a1 ==> 31

8) La función predefinida *unzip* tiene el siguiente comportamiento

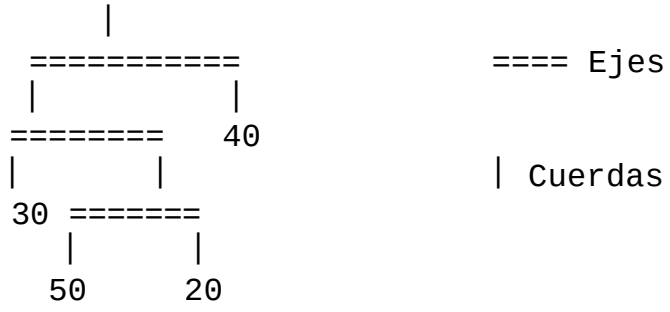
unzip [ ] ==> *([ ], [ ])*
unzip [(1,7), (3,4), (5,6)] ==> *([1,3,5] , [7,4,6])*

Define ésta función (dando su tipo) utilizando una función de plegado sobre listas (*1.5 punto*)

9) Para representar un *golgante* de los que suele haber en las habitaciones de los niños definimos el siguiente tipo:

type *Masa = Int*
type *Longitud = Int*
data *Colgante = Figura Masa | Eje Rama Rama* **deriving** *Show*
type *Rama =* (*Longitud, Colgante*)

Por ejemplo, el colgante



queda representado con la expresión

Eje (4 ,Eje (1, Figura 30) (7, Eje (5, Figura 50) (2, Figura 20))) (7, Figura 40)

a) Asumiendo que los ejes y las cuerdas del colgante no pesan, define una función *masa* :: *Colgante -> Masa* que calcule la masa total de un colgante (*1.25 puntos*)

b) Define una función *momento* :: *Rama -> Int* que calcule el momento ejercido por una rama (la longitud de la rama por su masa) (*0.5 puntos*)

c) Un colgante está balanceado si todos su ejes están horizontales. Para que un eje esté horizontal los momentos de sus dos ramas deben coincidir. Define una función *balanceado* :: *Colgante -> Bool* que indique si un colgante está balanceado. (*1.25 puntos*)