UNIVERSIDAD DE MÁLAGA
E.T.S.I. INFORMÁTICA

Programación Declarativa

(3º de Ingeniería Técnica en Informática)
Febrero de 2006

Nombre:
Especialidad: **Grupo:**

Haskell

Ejercicio 1

(a) ($1\ pt.$) Reescribe la función `g` con una lista por comprensión y da también su tipo polimórfico:

```
g x = map (aplicaA x)
aplicaA x f = f x
```

SOLUCION:

```
g :: a -> [a -> b] -> [b]
g x fs = [ f x | f <- fs ]
```

(b) ($0{,}5\ pt.$) Da el tipo polimórfico de la función `delete`:

```
delete x [] = []
delete x (y:ys) = if x == y then ys else y : delete x ys
```

SOLUCION:

```
delete :: Eq a => a -> [a] -> [a]
```

(c) ($0{,}5\ pts.$) Da el tipo polimórfico de la función `flip`:

```
flip f x y  = f y x
```

SOLUCION:

```
flip :: (a -> b -> c) -> b -> a -> c
```

(d) ($1\ pt.$) Da el tipo polimórfico del operador `(\\)`:

```
(\\) = foldl (flip delete)
```

SOLUCION:

```
(\\) :: Eq a => [a] -> [a] -> [a]
```

(e) ($1{,}5\ pts.$) Da el tipo polimórfico de la función `f`. Explica además qué hace:

```
f a = (a\\) . (a\\)
```

SOLUCION:

```
f :: Eq a => [a] -> [a] -> [a]
```

Comportamiento de f:

```
    f xs ys  devuelve los elementos de ys que están en xs.
```

(f) ($1\ pt.$) Considera las siguientes definiciones del Prelude:

```
data Maybe a = Nothing | Just a deriving (Eq, Ord, Read, Show)

lookup key [] = Nothing
lookup key ((x,y):xys)
  | key == x  = Just y
  | otherwise = lookup key xys
```

Define `f` y `z` para que la siguiente definición sea equivalente a la anterior:

SOLUCION:

```
lookup key = foldr f z
 where

   f (x,y) v = if key == x then Just y else v

   z = Nothing
```

(g) ($1{,}5\ pts.$) Define `f` y `z` para que la siguiente definición sea equivalente a la anterior:

SOLUCION:

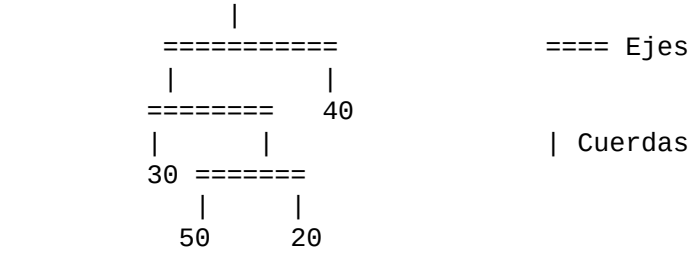
```
lookup' key = foldl f z
 where
   f (Just y) (x,_) = Just y

   f Nothing  (x,y) = if key == x then Just y else Nothing

   z = Nothing
```

Ejercicio 2

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) $(1\ pt.)$ 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.

SOLUCION:

```
masa :: Colgante -> Masa
masa (Figura m) = m
masa (Eje (l1,c1) (l2,c2)) = masa c1 + masa c2
```

(b) $(0{,}5\ pts.)$ Define una función `momento :: Rama -> Int` que calcule el momento ejercido por una rama (la longitud de la rama por su masa).

SOLUCION:

```
momento :: Rama -> Int
momento (l,c) = l * masa c
```

(c) $(1{,}5\ pts.)$) 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.

SOLUCION:

```
balanceado :: Colgante -> Bool
balanceado (Figura _) = True
balanceado (Eje r1@(l1,c1) r2@(l2,c2)) = momento r1 == momento r2
                                         && balanceado c1 && balanceado c2
```