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

Programación Declarativa

(3º de Ingeniería Técnica en Informática)
20 de febrero de 2003

Alumno: _____ Grupo: _____

1. Haskell

Ejercicio 1 (*3 puntos*)

- Escribe, usando las funciones `map` y `filter`, la función, dando además su tipo.

  ```
  k xs ys = [x+2 | x<-xs, elem x ys]
  ```

  Nota: la función `elem :: Eq a => a -> [a] -> Bool` comprueba si un dato pertenece a una lista.

- Escribe, usando lista por comprensión la función, dando además su tipo.

  ```
  k x xs = filter p (map (f x) xs)
  ```

- Escribe, usando `foldr` la función, dando además su tipo.

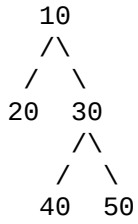
  ```
  k x xs = filter p (map (f x) xs)
  ```

Ejercicio 2 (*4 puntos*)

- Consideremos el siguiente tipo para representar árboles binarios

  ```
  data ÁrbolB a = VacíoB | NodoB (ÁrbolB a) a (ÁrbolB a)
  ```

  Define una función recursiva `enNivelB`, que tome un número y un árbol binario y devuelva una lista con todos los nodos del árbol cuyo nivel coincida con el indicado. Considérese que el dato en la raíz del árbol está a nivel 0 y que el nivel aumenta en uno cada vez que se desciende en el árbol.
  Por ejemplo, en el siguiente árbol

  el dato 10 está a nivel 0, los datos 20 y 30 a nivel 1 y los datos 40 y 50 a nivel 2. Si la variable `a` representa el árbol anterior, algunos ejemplos son

  ```
  enNivelB 0 a ==> [10]
  enNivelB 1 a ==> [20,30]
  enNivelB 2 a ==> [40,50]
  ```

  (No es necesario que los datos aparezcan en el orden indicado).

- Define una función recursiva `hastaNivelB`, que tome un número y un árbol binario y devuelva una lista con todos los nodos del árbol cuyo nivel no supere el indicado. Algunos ejemplos son

```
hastaNivelB 0 a ==> [10]
hastaNivelB 1 a ==> [10,20,30]
hastaNivelB 2 a ==> [10,20,30,40,50]
```

(No es necesario que los datos aparezcan en el orden indicado).

- Consideremos la siguiente función de orden superior:

```
pliegaB :: (a->b) -> (b->a->b->b) -> b -> (Integer -> ÁrbolB a -> b)
pliegaB f g z = fun
 where
  fun n VacíoB = z
  fun n (NodoB i r d)
   | n == 0    = f r
   | n > 0     = g (fun (n-1) i) r (fun (n-1) d)
```

Define las funciones `hastaNivelB` y `enNivelB` usando `pliegaB`.

Ejercicio 3 (*3 puntos*)

- Define una función `palabra :: (Char->Bool) -> String -> String` que extraiga la primera palabra de una lista de caracteres. La palabra estará formada por el mayor segmento inicial de caracteres que no sean separadores. El primer parámetro de la función `palabra` es una función que dado un caracter devuelve `True` si éste es un separador. Por ejemplo

```
palabra (==' ') "Esto es un texto" ==> "Esto"
palabra (\c -> c `elem` " ,.") "Esto, es un texto" ==> "Esto"
```

- Define una función `palabras :: (Char->Bool) -> String -> [String]` que extraiga todas las palabras de una cadena de caracteres. Por ejemplo

```
palabras (==' ') "Esto es un texto"
  ==> ["Esto","es","un","texto"]
palabras (\c -> c `elem` " ,.") "Esto, es un texto."
  ==> ["Esto","es","un","texto"]
```