Final Llenguatges de Programació

Grau en Enginyeria Informàtica
Temps estimat: 3h

17 de Gener de 2014

Es valorarà l'ús que es faci de funcions d'ordre superior predefinides i la simplicitat de la solució. Només s'han d'usar funcions de l'entorn Prelude. Si l'enunciat no ho impedeix, podeu usar les funcions auxiliars que us calguin.

Problema 1 (1.5 punts): *Parts*.
Definiu una funció en Haskell que donada una llista de nombres retorna la llista de llistes que són un subconjunt de la llista donada. No importa l'ordre en que apareixen les llistes en la llista resultant. Per exemple, per la llista `[3,5,8]` un resultat vàlid és la llista `[[],[8],[5],[5,8],[3],[3,8],[3,5],[3,5,8]]`.

Problema 2 (1.5 punts): *Coprimers*. Dos nombres enters positius (més grans que zero) són coprimers si no tenen cap factor primer en comú. Feu una funció en Haskell que genera la llista infinita en ordre creixent amb tots els coprimers d'un nombre donat.

Problema 3 (1.5 punts): *De generals a binaris*. Considereu les següents definicions per a arbres generals i arbres binaris.

```
data BTree a = BNode a (BTree a) (BTree a) | Abuit
data Tree a = Node a [Tree a]
```

Feu una funció `g2b :: Tree a -> BTree a` en Haskell que transforma un arbre general en binari (preservant el seu recorregut en preordre). Per això completeu les següents definicions:

```
g2b (Node x []) =
g2b (Node x [a]) =
g2b (Node x [a,b]) =
g2b (Node x (a:(Node y bs):as)) =
```

No podeu usar cap funció auxiliar ni canviar els casos.

Problema 3 (3.5 punts): *Inferència de tipus.* Cal escriure l'arbre decorat de les expressions i generar les restriccions de tipus. Resoleu-les per obtenir la solució.

1. Assumint que `(,) :: a -> b -> (a,b)` , inferiu el tipus més general de `fun1`:

   ```
   fun1 f (x,y) = f x y
   ```

2. Assumint que `(:) :: a -> [a] -> [a]` , inferiu el tipus més general de `fun2`:

   ```
   fun2 f x (y:ys) = let m = fun2 f x ys in if (f y) then (x:m) else (y:m)
   ```

3. Assumint que `0::Int`, `1::Int`, `+ :: Int -> Int -> Int` ,
   `fst :: (a,b) -> a` i que `(==) :: Eq a => a -> a -> Bool` , inferiu el tipus més general de `fun3`:

   ```
   fun3 x (y:ys) = if (fst y) == x then 0 else (fun3 x ys) + 1
   ```

Problema 4 (1.5 punts): *Python.*

1. Feu una funció en Python que donada una llista de strings `l` i un string `s` retorna una llista amb els strings de l que contenen s mantenint l'ordre relatiu en que es trobaven a `l`.

2. Feu una funció en Python que faci el mateix que el zip de Haskell. Per exemple `zip [2,3,4] [1,5]` retorna `[(2,1),(3,5)]`. Recordeu que la llista resultant té la mida de la llista més curta de les dues donades. Recordeu que `range(0,n)` genera la llista de tots els nombres entre `0` i `n`.

Problema 5 (0.5 punt): *Conceptes de llenguatges de programació.*

1. Indiqueu què significa que el "tipat" en un llenguatge de programació sigui estàtic o dinàmic.

2. Indiqueu si el tipat és dinàmic o estàtic en el llenguatge que us va tocar en el Treball Dirigit (TD) de Competències Transversals.