Examen final de Llenguatges de Programació

Grau en Enginyeria Informàtica
Temps estimat: 2h i 50m
15 de gener de 2016

Els Problemes 1, 2 i 3 s'han de resoldre amb Haskell usant només l'entorn Prelude. Es valorarà l'ús que es faci de funcions d'ordre superior predefinides i la simplicitat de la solució.

Problema 1 (1.25 punts): *Llistes periòdiques.* Feu una funció `periodic :: [Int] -> [Int]`, que donada una llista d'entrada genera la llista (possiblement infinita) que conté tots els elements de la llista d'entrada seguits per els elements la segona meitat (per excés) de la llista d'entrada repetits indefinidament. Noteu que la llista d'entrada pot ser infinita i que en aquest cas, el resultat és la mateixa llista. Com a exemples tenim que

```
periodic [2,8,5] = [2,8,5,8,5,8,5,...
periodic [2,8,5,2,1,6] = [2,8,5,2,1,6,2,1,6,2,1,6,...
periodic [1] = [1,1,1,1,1,...
periodic [2..] = [2,3,4,5,6,7,8,9,10,11,...
```

Problema 2 (1.25 punts): *Llistes transposades.* Feu una funció `transpose :: [[a]] -> [[a]]`, que calcula la transposada de la llista original. És a dir, que a la llista resultat la primera llista està formada pels primers elements de les llites que conté l'entrada (en el mateix ordre), la segona pels segons, i així fins que alguna de les llistes de la llista d'entrada no tingui més elements. Per exemple, `transpose [[2,5,6,1],[8,8,5],[3,14,1,2,5]] = [[2,8,3],[5,8,14],[6,5,1]]`

Problema 3 (2 punts): *QuadTrees.* Volem definir un tipus arbre per representar *QuadTrees*, que permeten emmagatzemar conjunts d'elements que es poden organitzar en quatre quadrants. Per exemple, els punts en dos dimensions són així. D'aquesta manera, si el QuadTree està equilibrat, operacions com la inserció o la cerca es poden fer eficientment (en temps logarítmic). La idea és la mateixa que a la pràctica obligatòria de Haskell. Per aconseguir la màxima genericitat, crearem una `class` que representi elements que es poden repartir en quadrants. Es demana

1. Definiu la `class Quadable` que conté les operacions

   - `quad`, que rep dos elements x i y del tipus de la classe i retorna un enter entre 0 i 3 que indica a quin quadrant està y respecte x.
   - `in1st`, `in2nd`, `in3rd` i `in4th`, que reben dos elements x i y del tipus de la classe i retornen un booleà, que indiquen respectivament si y respecte x està en el primer, el segon, el tercer o el quart quadrant.

   Definiu (dins de la classe) `quad` usant `in1st`, `in2nd`, `in3rd` i `in4th`, i també `in1st`, `in2nd`, `in3rd` i `in4th` usant `quad`.

2. Defineix el tipus `PointInt2D` que representa punts en dos dimensions formats per dos enters. Feu que `PointInt2D` sigui instance de la classe `Quadable`

3. Defineix el tipus polimòrfic `QuadTree`. Per a poder emmagatzemar un conjunt de punts en un arbre amb quatre fills. Per exemple, pels punts $(2,2)$, $(-1,0)$, $(-3,2)$, $(1,3)$, $(5,0)$, $(-3,-2)$, $(0,1)$ un possible `QuadTree` seria

   ```
   Node (Point 0 1) [Node (Point 1 3) [Empty,Empty,Empty,Node (Point 2 2) [Empty,Empty,Empty,Empty]],
                     Node (Point (-3) 2) [Empty,Empty,Empty,Empty],
                     Node (Point (-3) (-2)) [Node (Point (-1) 0) [Empty,Empty,Empty,Empty],Empty,Empty,Empty],
                     Node (Point 5 0) [Empty,Empty,Empty,Empty]]
   ```

4. Definiu l'operació `insert :: Quadable a => a -> QuadTree a -> QuadTree a` que insereix un nou element en el `QuadTree` de manera que a cada element el podem trobar per una única branca seguint la informació dels quadrants.

5. Definiu l'operació `member :: (Eq a, Quadable a) => a -> QuadTree a -> Bool` que indica si un element donat està en el `QuadTree`.

Problema 4 (2.5 punts): *Inferència de tipus.* Cal escriure l'arbre decorat de les expressions i generar les restriccions de tipus. Resoleu-les per obtenir la solució. Assenyaleu el resultat final amb un requadre.

1. Tenint en compte que `fst :: (a,b) -> a` i `snd :: (a,b) -> b`, inferiu el tipus més general de `fun1`:

   ```
   fun1 f x = f (fst x) (snd x)
   ```

2. Tenint en compte que `(,) :: a -> b -> (a,b)` , `(:) :: a -> [a] -> [a]` i que `(==) :: Eq a => a -> a -> Bool` , inferiu el tipus més general de `fun2`:

   ```
   fun2 ((x,y):xs) = let r = (fun2 xs) in if x==y then x:r else r
   ```

Problema 5 (2.5 punts): *Python.*

1. Considereu la següent definició de la classe `BTree` que es dona al final de l'exercici i que implementa els arbres binaris no buits. Definiu una subclasse `SBTree` de la classe `BTree`, que afegeixi l'operació `getSize` que retorna la mida (és a dir, el nombre d'elements) de l'arbre. Heu d'afegir un atribut `size` que manté aquesta mida i modificar les operacions que calgui de manera que totes les operacions tinguin cost constant.

2. Implementeu la funció `minOccur` que donat un `SBTree` retorna una tupla de dos elements on la primera component és l'element del `SBTree` que menys cops apareix i la segona quants cops hi apareix (si no és únic torneu un qualsevol d'ells)

```
class BTree:                          |     # exemple de crides i resultat
    def __init__ (self, x):           |     a = SBTree(8)
       self.rt = x                    |     a1 = SBTree(5)
       self.left = None               |     a2 = SBTree(8)
       self.right = None              |     a3 = SBTree(3)
    def setLeft (self, a):            |     a4 = SBTree(8)
        self.left = a                 |     a5 = SBTree(3)
    def setRight (self, a):           |     a6 = SBTree(5)
        self.right = a                |     a5.setRight(a6)
    def root (self) :                 |     a3.setRight(a4)
        return self.rt                |     a3.setLeft(a5)
    def leftChild (self)              |     a1.setLeft(a3)
        return self.left              |     a1.setRight(a2)
    def rightChild (self)             |     a.setLeft(a1)
        return self.right             |     print a.getSize()
                                      |     7
                                      |     print minOccur(a)
                                      |     (3,2)
```

Problema 6 (0.5 punts): *Conceptes de llenguatges de programació.*

1. Indiqueu com és el tipat de Python (fort/feble i estàtic/dinàmic). És el tipat de Python com el de la majoria dels llenguatges de scripting?

2. Indiqueu quin és el llenguatge que heu fet al Treball Dirigit (TD) de Competències Transversals i com és el seu tipat.