Examen final de Llenguatges de Programació

Grau en Enginyeria Informàtica
Temps estimat: 2h i 45m

8 de gener de 2015

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.

Problema 1 (1.75 punts): *Mínim comú múltiple.*

1. Feu una funció `lmult`, que genera la llista (infinita) ordenada creixentment amb tots el múltiples d'un enter donat. Per exemple, `lmult 7` és `[7,14,21,28,... ]`.

2. Usant la funció anterior, feu una funció `mcm`, que donats dos enter positius obté el mínim comú múltiple dels dos.

Problema 2 (1.75 punts): *Llistes infinites periòdiques.* Volem definir un nou tipus de dades genèric `PList` que representi les llistes infinites que segueixen un període (de forma similar als nombres racional). Aquestes llistes estan formades per dues parts, l'*inici* i el *període*. Així, per llistes d'enters, si l'inici és `[1,6,3]` i el període `[-3,8]`, tenim la llista infinita `[1,6,3,-3,8,-3,8,-3,8, ...]`

1. Definiu el data `PList` per representar aquestes llistes infinites periòdiques on el tipus dels elements de la llista és genèric i on la llista de l'exemple anterior s'obté amb l'expressió `(Period [1,6,3] [-3,8])`. És a dir, usant el constructor `Period`.

2. Feu la funció `generate :: PList a -> [a]` que, donada una llista infinita periòdica, genera la llista infinita que representa. Per exemple,

   `generate (Period [1,6,3] [-3,8])` és `[1,6,3,-3,8,-3,8,-3,8, ...]`

   `generate (Period ['a'] ['b','a'])` és `['a','b','a','b','a','b','a', ...]`

3. Dues llistes infinites es consideren iguals, si tots els prefixos de la mateixa longitud són iguals. Indiqueu que PList és instance de la classe Eq on (==) és la igualtat de les llistes infinites que representen. Per exemple,

   `(Period ['a','b'] ['a','b'])` i `(Period ['a'] ['b','a'])` són iguals, però diferents de `(Period ['a'] ['b','a','b'])`

   Per a fer la comprovació, afortunadament no cal considerar tots els possibles prefixos, sinó només el que té com a mida las suma del màxim de les longituds dels inicis més el mínim comú múltiple de les longituds dels períodes. Per exemple, per `(Period ['a','b'] ['a','b'])` i `(Period ['a'] ['b','a','b'])`, tenim que el màxim de les longituds dels inicis és 2 i el mínim comú múltiple de les longituds dels períodes és 6. Per tant, cal considerar el prefix de longitud 8. Aquesta longitud no és òptima, però és una aproximació raonable.

Problema 3 (1 punt): *Nivells d'arbres.* Considereu els arbres generals **no buits**, representats amb un únic constructor `Node`.

1. Definiu un tipus polimòrfic `Tree` per construir arbres com ara

```
Node 2 [Node 5 [Node 3 [Node 8 []],Node 6 [Node 2 [],Node 4 []],Node 1 []],Node 3 [Node 8 []],Node 4 [Node 5 [],Node 1 [],Node 2 []]]
```

2. Feu una funció `nivell :: Int -> Tree a -> [a]`, tal que donat un enter n i una arbre general, ens retorna la llista d'elements que estan a nivell (o profunditat) n, considerant que l'arrel es troba a profunditat 0. Per exemple, per l'arbre de l'apartat anterior el resultat a nivell 2 és `[3, 6, 1, 8, 5, 1, 2]`.

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 `(:) :: a -> [a] -> [a]` i `length :: [a] -> Int`, inferiu el tipus més general de `fun1`:

   ```
   fun1 (x:l) = let n = length x in n:(fun1 l)
   ```

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

   ```
   fun2 f x y = f x == fst y
   ```

Problema 5 (2.5 punts): *Python.*

1. La profunditat de llista d'un objecte és el nivell màxim d'imbricament de llistes, entenent que qualsevol objecte que no sigui de la classe llista té profunditat zero. Per exemple, `3` té profunditat 0, la llista `[3,4]` té profunditat 1, la llista `[[6]]` té profunditat 2 i la llista `[[5,[]],[]]` té profunditat 3. Feu una funció en Python que donada una llista `l` retorna un diccionari indexat per profunditat que ens indica quants elements de la llista hi ha per a cada profunditat. No s'han d'incloure les profunditats que no tenen cap element. Per exemple, per la llista `[[3,4],[[6]],[[5,[]],[]],[[1,2],[]],3]` el resultat és {`0:1,1:1,2:2,3:1`}.

2. Considereu la següent definició incompleta de la classe `Tree` que es dona al final del exercici i que ha d'implementar els arbres generals no buits. Completeu l'operació `__init__` de classe Tree, l'operació `ithChild` que retorna l'ièssim (el primer és el zero) fill de l'arbre i l'operació `numChildren` que retorna el nombre de fills de l'arbre.

   Definiu una subclasse `Pre` de la classe `Tree`, que afegeixi l'operació `preorder` que retorna una llista amb el recorregut en preordre de l'arbre al que s'aplica.

```
class Tree:
    def __init__ (self, x):          |      # exemple de crides i resultat
        ...                          |      a = Pre(2)
    def addChild (self, a):          |      a.addChild(Pre(3))
        self.child.append(a)         |      a.addChild(Pre(4))
    def root (self) :                |      print a.preorder()
        return self.rt               |
    def ithChild ...                 |      [2, 3, 4]
        ...                          |
    def numChildren ...              |
        ...
```

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

1. Indiqueu quin llenguatge heu fet al Treball Dirigit (TD) de Competències Transversals.

2. Indiqueu si aquest llenguatge és de scripting i si és compilat o interpretat.

3. Indiqueu què significa que el "tipat" en un llenguatge de programació sigui estàtic o dinàmic, i quin és el cas del vostre llenguatge.