Examen final de Llenguatges de Programació

Grau en Enginyeria Informàtica
Temps estimat: 2h i 50m
10 de gener de 2019

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 l'eficiència i la simplicitat de la solució.

Si veu treure més d'un 4 del l'examen parcial de Haskell (sense comptar la pràctica), podeu optar per no respondre aquests tres problemes, i en aquest cas la nota d'aquests tres problemes (4.5) serà 4.5*NotaParcialHaskell/10.

Problema 1 (1.5 punts): *Llistes.* Feu una funció `genNumLists:: [[Integer]]`, que genera la llista infinita que conté en ordre creixent una llista infinita repetint el mateix enter no negatiu. És ha dir, `[[0,0,0,...],[1,1,1,...],[2,2,2,...],...]`.

Usant la funció anterior feu la funció `squareList:: Int -> [[Integer]]`, que donat un enter no negatiu k retorna la llista amb k llistes de mida k amb $[[0,\ldots,0],...,[k-1,\ldots,k-1]]$. Per exemple,

```
squareList 0 = []
squareList 5 = [[0,0,0,0,0],[1,1,1,1,1],[2,2,2,2,2],[3,3,3,3,3],[4,4,4,4,4]]
```

Problema 2 (1 punt): *Comptant.* Feu una funció `counter:: Eq a => [a] -> [a] -> [(a,Int)]` que rep dues llistes l_1 i l_2 i retorna una llista de parells on la primera component és un element de la llista l_2 i la segona component és quants cops apareix aquest element en la llista l_1. Els parells han d'apareixer en el mateix ordre relatiu (respecte a la primera component) que en la llista l_2. Per exemple:

```
counter [3,4,5,2,2,3,4,1,5,5,2] [2,5,3] = [(2,3),(5,3),(3,2)]
counter [3,4,5,2,2,3,4,1,5,5,2] [0..5] = [(0,0),(1,1),(2,3),(3,2),(4,2),(5,3)]
```

Problema 3 (2 punts): *BSTList.* Considereu arbres binaris de cerca, on

- als nodes tenim com sempre elements, i dos arbres de cerca on tots els elements que apareixen a l'arbre de l'esquerra són estricatment més petits que l'element del node i tots els elements que apareixen a l'arbre de la dreta són estricatment més grans que l'element del node.
- a les fulles tenim llistes d'elements (no repetits).

Feu els següents apartats:

1. Per a representar aquests arbres en Haskell definiu el `data` polimòrfic `BSTList a` amb constructors `Node` i `List`. Com a dos exemples de valors d'aquest data teniu:

   ```
   Node 10 (Node 5 (List [3,-2,4]) (List [8,6])) (Node 17 (List [12,14]) (List [21,18,27]))
   Node 'm' (Node 'f' (List ['a','d']) (List [])) (List [])
   ```

2. Feu una funció `isIn :: Ord a => a -> BSTList a -> Bool`, que donat un valor x i un BSTList b, ens diu si x apareix en b. És molt important evitar recorrer nodes innecessaris del BSTList.

3. Feu una funció `insert :: Ord a => Int -> a -> BSTList a -> BSTList a` Que donat un enter no negatiu k, un valor x i un BSTList b (que podeu assumir que no conté cap llista de longitud superior a k) ens retorna un BSTList resultant d'afegir x a b i que no té cap llista de longitud superior a k. Recordeu que el BSTList no ha de tenir elements repetits i que no heu de visitar nodes innecesaris. Per exemple, si fem `insert 3 1 ej1` on `ej1` és al primer arbre anterior obtenim:

   ```
   Node 10 (Node 5 (Node 1 (List [-2]) (List [3,4])) (List [8,6])) (Node 17 (List [12,14]) (List [21,18,27]))
   ```

   i si fem `insert 3 15 ej1`, obtenim:

   ```
   Node 10 (Node 5 (List [3,-2,4]) (List [8,6])) (Node 17 (List [15,12,14]) (List [21,18,27]))
   ```

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 -> b -> (a,b)` and `snd :: (a,b) -> b`, inferiu el tipus més general de `fun1`:

   ```
   fun1 x (y,z) f = let e = snd x in f e y
   ```

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):l) = (x==y) :  (fun2 l)
   ```

Problema 5 (2.5 punts): *Python.*

1. Feu una funció Python3 `exists`, que donat un diccionari retorna una funció que donat un valor ens diu si és una clau del dicionari. Així, si fem

   ```
   f = exists({3:"a",2:"q",13:"o"})
   print(f(5))
   print(f(3))
   ```

   escriu primer `False` i després `True`.

2. Considereu una classe per representar els BSTList del Problema 3, però en Python 3. Feu l'implementació de la funció `insert`, que fa el mateix que la de l'apartat 3 del Problema 3. La vostra implementació de la funció insert ha de ser compatible amb la resta de la implementació de la class que us donem a continuació i els resultats que es mostren.

   ```
   class BSTList:                                  |  # exemple de crides i resultat
     def __init__ (self,l):                        |  a=BSTList([])
       self.lval = l                               |  a.insert(2,8)
                                                   |  a.printB("\n")
     def insert (self,k,x):                        |  List([8])
         .                                         |  a.insert(2,23)
         .                                         |  a.printB("\n")
         .                                         |  List([8, 23])
                                                   |  a.insert(2,9)
     def printB (self,f=" "):                      |  a.printB("\n")
       if hasattr(self,"lval"):                    |  Node(9 List([8]) List([23]) )
         print("List("+str(self.lval)+")", end=f)|  a.insert(2,15)
        else:                                      |  a.printB("\n")
          print("Node("+str(self.n), end=" ")     |  Node(9 List([8]) List([23, 15]) )
          self.left.printB()                       |  a.insert(2,13)
          self.right.printB()                      |  a.printB("\n")
          print(")", end=f)                        |  Node(9 List([8]) Node(13 List([]) List([23, 15]) ) )
   ```

   Recordeu que per eliminar un atribut d'un objecte disposeu de l'operació `delattr` que rep com a paràmetres el objecte i el nom del atribut (com a string).

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

1. Indiqueu les propietats del sistema de tipus de Haskell.

2. Indiqueu quin llenguatge us va tocar en el Treball Dirigit (TD) de Competències Transversals, si és o no de scripting i quins paradigmes inclou. Si té extensions importants, indiqueu quins paradigmes incorporen.