Examen final de Llenguatges de Programació

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

10 de juny 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 (1.5 punts): *Successió de Thue-Morse* (Haskell).
La successió de Thue-Morse és una successió infinita $d_0, d_1, \ldots$ de zeros i uns, que satisfà que $d_0 = 0$ i $d_{2n} = d_n$ i $d_{2n+1} = 1 - d_n$ per tot $n \geq 0$.

Definiu la llista `thuemorse :: [Int]` com la llista infinita dels elements de la successió, és a dir, `[0,1,1,0,1,0,0,1,1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0,0,1,...`

Problema 2 (2.5 punts): *BST estès* (Haskell).
Seguint la idea de que el *quicksort* es comporta millor si els subvectors de mida petita s'ordenen per inserció, volem definir un nou tipus de dades genèric basat en els BST (*Binary Search Tree*) que admet que a les fulles hi hagi llistes ordenades. Aquest tipus que anomenarem EBST, té dos constructors: un pels nodes interns (que són com en un arbre binari) i un per les fulles que contenen llistes (que poden ser buides).

Assumirem que un EBST sempre compleix que tots els elements de l'esquerra d'un node intern (inclosos els que són a les llistes) són menors que ell i tots els de la dreta (inclosos els que són a les llistes) són més grans o iguals que ell. A més, les llistes de les fulles estan ordenades.

Considereu com exemple el següent EBST:

```
Split 8 (Split 1 (Leaf []) (Split 6 (Split 4 (Leaf [2,3]) (Leaf [4,5])) (Leaf []))) (Leaf [9,10])
```

1. Definiu el data polimòrfic `EBST` que permeti representar els arbres que s'han descrit.

2. Definiu la funció `addEBST :: Ord a => a -> EBST a -> Int -> EBST a`

   que donat un element x, un EBST a i un enter n que indica la mida màxima de les llistes del EBST, insereixi ordenadament x en a sense que cap llista de l'arbre resultant tingui més de n elements. L'algorisme no ha de recórrer nodes innecessaris i no cal que l'arbre quedi equilibrat en cap sentit.

3. Feu que `EBST` sigui instance de la classe `Eq` on dos `EBST` són iguals si contenen els mateixos elements com a conjunt, és a dir, sense tenir en compte els repetits. Es valorarà l'eficiència de la solució.

Problema 3 (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]`, inferiu el tipus més general de `fun1`:

   ```
   fun1 f x = x:(fun1 f (f x))
   ```

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

   ```
   fun2 x l = let u = map fst l in map (+x) u
   ```

Problema 4 (3 punts): *Python.*

1. Feu una funció `classify` que rep un objecte Python i troba tots els strings que hi ha travessant només llistes o tuples i ens retorna un diccionari que per a cada freqüència d'aparició ens indica els string trobats que la tenen. Si una freqüència no te cap paraula associada no ha d'aparèixer al diccionari. Per exemple, amb

   `['dia',(2,"hola"),[3,"hola",3],'dia',5,(3.6,"mes"),("dia",1),["hotel",2,("hola"),{1:'hola'}]]`
   el resultat és `{1:['mes','hotel'],3:['dia','hola']}`

   Noteu que l'últim `'hola'` que apareix no es té en compte perquè apareix en un diccionari. A més, noteu que el resultat seria el mateix si qualsevol llista la canviem per una tupla (i al reves).

2. Considereu la següent definició incompleta de la classe `Poly` que es dona al final del exercici i que ha d'implementar els polinomis sobre una variable. Completeu l'operació `__init__` de classe Poly i l'operació `add_coeff` que afegeix un coeficient d'un grau determinat. Noteu que `get_degree` retorna el grau del polinomi. Assumiu en aquest exercici que el polinomi buit és de grau zero.

   Definiu una subclasse `PolyEval` de la classe `Poly`, que afegeixi l'operació `eval` que avalua el polinomi en el punt que ens passen per paràmetre.

```
class Poly:                                 |     # exemple de crides i resultat
    def __init__(self):                     |     p = PolyEval()
        ...                                 |     p.add_coeff(1,2)
    def add_coeff(self,degree,coeff):       |     p.add_coeff(3,3.5)
        self.coeffs[degree] = coeff         |     p.add_coeff(0,3)
        ...                                 |     print p.get_degree()
    def get_degree(self):                   |     print p.eval(2)
        return self.degree                  |
                                            |     3
                                            |     35.0
```

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

1. Indiqueu quin llenguatge heu fet al Treball Dirigit (TD) de Competències Transversals.
2. Indiqueu quins paradigmes admet i si és compilat o interpretat.
3. Indiqueu si hi ha alguna relació entre que el tipat sigui estàtic o dinàmic i el fet de que el llenguatge sigui compilat o interpretat.