Examen final de Llenguatges de Programació

Grau en Enginyeria Informàtica
Temps estimat: 2h i 50m
16 de gener de 2017

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.

Problema 1 (1 punt): *addEachOne*. Feu una funció `addEachOne :: [a] -> [[a]] -> [[a]]`, que rep una llista i una llista de llistes i retorna la llista de les llistes obtingudes afegint a cada llista del segon paràmetre un element de la llista del primer paràmetre com a primer element. El resultat ha de mantenir l'ordre relatiu de les llistes d'entrada com es mostra als exemples:

```
addOneEach [3,7] [[2,1],[9]] = [[3,2,1],[3,9],[7,2,1],[7,9]]
addOneEach [2,8,9] [[]] = [[2],[8],[9]]
```

Noteu que primer posem el primer element de la primera llista a totes les de la segona, després el segon, etc.

Problema 2 (1 punt): *generateAll*. Feu una funció `generateAll :: [Int] -> [[Int]]`, que donada una llista d'enters que suposarem ordenada i sense repetits, genera la llista infinita amb totes les llistes formades amb elements de la llista d'entrada. Han d'aparèixer ordenades per mida i entre les de la mateixa mida apareixeran ordenades lexicogràficament (comparant el enters de la llista d'esquerra a dreta).

```
generateAll [3,5,7]
[[],[3],[5],[7],[3,3],[3,5],[3,7],[5,3],[5,5],[5,7],[7,3],[7,5],[7,7],[3,3,3],[3,3,5],[3,3,7],
[3,5,3],[3,5,5],[3,5,7],[3,7,3],[3,7,5],[3,7,7],[5,3,3],[5,3,5],[5,3,7],[5,5,3],[5,5,5],[5,5,7],
[5,7,3],[5,7,5],[5,7,7],[7,3,3],[7,3,5],[7,3,7],[7,5,3],[7,5,5],[7,5,7],[7,7,3],[7,7,5],[7,7,7],...
```

Problema 3 (2.5 punts): *Arbres binaris i heaps equilibrats*. Volem treballar amb un heap (que emmagatzema elements parcialment ordenats creixentment) usant arbres binaris ampliats. Recordeu que en un heap els elements que apareixen en cada branca estan ordenats creixentment des de l'arrel a la fulla (però no tenim cap relació entre els elements de diferents branques), és a dir, per a tot subarbre els elements dels nodes dels fills són més grans que l'element del node pare. Es demanen els següents apartats.

1. Definiu un data polimòrfic `SBTree a` per representar arbres binaris (ampliats) d'un tipus genèric on a cada node tenim un `Int`, un element del tipus genèric i dos arbres com a fills. Com a exemple (particularitzant el tipus genèric `a` com `Double`) tenim:

   ```
   sh1 = SNode 9 3.2 (SNode 4 4.1 (SNode 1 14.0 Empty Empty)
                                  (SNode 2 6.1 Empty (SNode 1 9.3 Empty Empty)))
                     (SNode 4 8.1 (SNode 2 10.2 (SNode 1 11.5 Empty Empty) Empty)
                                  (SNode 1 16.0 Empty Empty))
   ```

2. Feu que `(SBTree a)` sigui `instance` de la classe `Eq` considerant que dos arbres són iguals si son idèntics si obviem el segon valor de cada node (en l'exemple els `Double`). En aquest apartat no podeu assumir res sobre el valor `Int` de cada `SNode`.

3. Per a aquest apartat usarem el valor `Int` del `SBTree a` per a tenir la mida (nombre de nodes) de l'arbre que comença en aquell node. Així usarem un `SBTree a` per a emmagatzemar heaps d'elements de tipus `a` que siguin equilibrats en el sentit de que els dos fills de cada subarbre tenen com a molt diferència 1 en la mida. L'exemple anterior és un heap de reals equilibrat i amb la mida per a cada subarbre no buit.

   Feu la funció `hinsert :: Ord a => a -> SBTree a -> SBTree a`, que donat un element i un heap equilibrat representat en un SBTree equilibrat (i amb mides) com s'ha descrit anteriorment, retorna un nou heap equilibrat (i amb mides) que conté també aquest element. Si l'element ja hi és, no s'ha d'afegir.

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)`, inferiu el tipus més general de `fun1`:

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

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

   ```
   fun2 x (y:l) = let s = sum l in if s < x then fun2 x l else y:l
   ```

Problema 5 (2.5 punts): *Python.*

1. Feu una funció Python `evaluate(e)`, que avalua el paràmetre `e` que rep considerant que els elements d'una llista es multipliquen, els de les tuples es sumen i els enters tenen el seu valor. La resta de tipus s'han d'ignorar. La llista buida avalua a 1 i la tupla buida a 0. Així `evaluate([3,(2,[8,"sol"],{}),7,([2,(1,1)],11)])` retorna 3150 i `evaluate(([(),7],[]))` retorna 1

2. Una funció de Python pot rebre un nombre variable de paràmetres mitjançant un paràmetre formal especial prefixat per * que contindrà la llista dels paràmetres reals.

   Per exemple, la funció `s` següent rep un nombre variable de paràmetres i en retorna la seva suma, de manera que `s(5,2)` retorna 7, s(3,1,2) retorna 6, i s() retorna 0:

   ```
   def s(*xs):
       m = 0
       for x in xs: m += x
       return m
   ```

   De manera semblant, es pot invocar una funció amb certs paràmetres formals a través d'un sol paràmetre real de tipus llista. Per exemple, si es té la funció `s2` següent que retorna la suma dels seus dos paràmetres,

   ```
   def s2(a,b):
       return a+b
   ```

   llavors es pot invocar-la amb un sol paràmetre de tipus llista tot prefixant la llista amb *: `s2(*[2,3])` retorna 5 perquè és equivalent a `s2(2,3)`.

   Es demana que implementeu la funció `partial(f, x)` que, donada una funció `f` d'un o més paràmetres i un paràmetre `x`, retorni una nova funció g tal que g sigui l'aplicació parcial (com amb les funcions currificades) de `f` amb el seu primer paràmetre valent `x`.

   Per exemple, amb la funció `s2` anterior, quan es fa `h = partial(s2, 2)` llavors `h(3)` retorna 5. Igualment, amb la funció `s` anterior, quan es fa `ff = partial(partial(s, 3), 1)` llavors `ff(2)` retorna 6.

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

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

2. Indiqueu quin llenguatge us va tocar en el Treball Dirigit (TD) de Competències Transversals, quines propietats té el seu sistema de tipus i quins paradigmes inclou. Si té extensions importants, indiqueu quins paradigmes incorporen.