Parcial Llenguatges de Programació

Grau en Enginyeria Informàtica

30 Novembre 2016

Es valorarà l'ús de funcions d'ordre superior predefinides. Ara bé, només es poden usar funcions predefinides de l'entorn Prelude: no podeu fer cap `import`.
Problema 1 (1.5 punt): *Suma de quadrats.* Implementeu les funcions següents:
Apartat 1.1. Feu la funció `quadrats::[Integer]` que genera la llista de tots els quadrats perfectes: 1, 4, 9, 16, 25, 36, ...
Apartat 1.2. Usant l'anterior funció, feu la funció `sumQuadrats::Integer -> Bool` que ens diu si un número es suma d'n quadrats perfectes consecutius. Exemples: el 13, 14 i el 77 ho són. El 3, 10 i el 26 no ho són.

Problema 2 (1.5 punt): *Serie de Conway.* Feu la funció `conway::[Integer]` que genera la serie 1, 1, 2, 2, 3, 4, 4, 4, 5, 6, 7, 7, ... on el primer i el segon elements a_1 i a_2 són 1 i la resta s'obtenen mirant la posició que indica l'últim element generat i sumant els dos element que em torbo en la posició que indiqui aquest nombre començant pel principi i pel final. Per exemple, a_9 és 5 per que a_8 és 4 i el quart element començant pel principi és 2 i el quart començant pel final (fins el a_8) és 3.

Problema 3 (3.5 punts): *Divideix i vençe.* Heu de crear les operacions següents:

Apartat 3.1: Una operació d'ordre superior
`dc::(a -> Bool) -> (a -> b) -> (a -> [a]) -> (a -> [b] -> b) -> a -> b`
que donades (1) una funció que indica si estem en un cas *trivial*, (2) una funció que *resol* els casos trivials, (3) una funció que *parteix* un problema en una llista de subproblemes, (4) una funció que *combina* el problema i la llista de solucions dels subproblemes per obtenir la solució i (5) un *problema*, resol el problema mitjançant l'esquema de *divideix i vençe*. És a dir,

```
dc trivial resol parteix combina problema
```

Noteu que en general `parteix` ha de retornar una llista, encara que en la majoria de casos és parteix només en dues parts. Igualment, noteu que en general, l'operació `combina` usa les solucions als subproblemes, però també tot o part del problema original.

Apartat 3.2: Utilitzant l'operació `dc`, creeu la funció
`quicksort::Ord a => [a] -> [a]`
que implementi l'algorisme d'ordenació quicksort. Per això, heu de definir totes les funcions auxiliars que us calguin per a fer la crida a `dc`.

Problema 4 (3.5 punts): *Aplanar*. Definiu un data `GTree` per expressar arbres generals de la forma

```
Node "a" [Node "b" [Node "b" [Node "d" [], Node "e" []],Node "c" [],Node "c" []], Node "a" []]
```

On el contingut dels nodes no ha de ser necessàriament strings.

Apartat 4.1. Definiu definiu la funció `flat` que aplana l'arbre de manera que si un fill té la mateixa arrel que el pare, el fill es substitueix pels seus fills. A l'exemple, ens queda

```
Node "a" [Node "b" [Node "d" [],Node "e" [],Node "c" [],Node "c" []]]
```

Després d'aplanar cap fill pot tenir la mateixa arrel que el seu pare.

Apartat 4.2. Feu que el tipus `GTree` sigui `instance` de la classe `Eq` on (==) és la igualtat dels arbres després d'aplanar i on l'ordre dels fills no importa.