Laboratori de PL: Haskell. Sessió 2

1 Compilació separada. Més Entrada/Sortida

Considerem ara la compilació separada. Mireu el codi del directori `codi/ARB`. Crideu al compilador amb

```
ghc opsArbre.hs
```

En aquest cas, també ens genera un executable anomenat `opsArbre`, però, si mireu el que hi ha al directori, veureu que també ha compilat el codir de l'arxiu `Arbre.hs`. També podrieu haber compilat primer els dos arxius generant els objectes amb

```
ghc -c Arbre.hs
ghc -c opsArbre.hs
```

i finalment muntan amb

```
ghc opsArbre.o Arbre.o
```

Cosa que gnera un executable anomenat `a.out`. De tota manera, la primera versió és molt més aconsellable ja que, a més, fa gestió de versions. Per a comprovar-ho feu

```
touch opsArbre.hs
ghc opsArbre.hs
```

i veureu com només compila opsArbre.hs

Per a que tingueu un altre exemple de com llegir l'entrada en diferents línias però sense haver de posar salts de línia entre cada dada, mireu el fitxer `lectura2.hs`

En aquest arxiu també es mostra com usar el `read` per a convertir un string en Int un cop hem convetit l'entrada en una llista de strings.

Noteu que en aquest exemple la lectura està feta en dos fases per a que us sigui més fàcil de reaprofitar, però en realitat es podria fer en una sola fase fent que `readall` generi directament una llista de llistes.

2 Funcions d'ordre superior

Una funció d'ordre-superior és una funció que, o pren una funció com argument o retorna una funció com a resultat.

En Haskell totes les funcions que definim que tinguin un o més paràmetres poden ser vistes com a funcions d'ordre superior, ja que si no els hi apliquem tots els paràmetres ens retornen una funció. Noteu que les dues declaracion següents són equivalents:

```
mesk:: Int -> (Int -> Int)

mesk:: Int -> Int -> Int
```

És a dir, `->` té associativitat per la dreta.

En general, en els llenguatges funcionals, les funcions també són considerades dades, i per això és tan natural tenir funcions d'ordre superior.

En aquesta sessió considerarem les funcion d'ordre-superior predefinides en haskell i veurem les seves aplicacions.

3 Funcions d'ordre superior predefinides més comunes en Haskell

Una de les funcions més conegudes és el `map` que aplica una funció a tots els elements d'una llista.

```
map :: (a->b) -> [a] -> [b]
map f  [] = []
map f (x:xs) = f x : map f xs
```

Altres funcions d'ordre superior força utilitzades són: `foldr`, `foldl`, `iterate`, `all`, `any`, `filter`, `dropWhile`, `takeWhile`, `zip`, `zipWith`.

En particular el `foldl` exiteix en molts llenguatges de programació, com ara a C++, Python, JavaScript, etc, amb diferents noms i, potser aplicats a arrays o altres contenidors.

Mireu l'arxiu `codi/funcionsOS.hs`. Ompliu el codi de les operacions que falten. Per provar-les canvieu lleugerament el nom de les funcions.

4 Ús de funcions d'ordre superior predefinides

Proveu l'ús d'aquestes funcions amb diferents exemples entrant a `ghci`.

Feu les següents funcions utilitzant les funcions d'ordre superior.

1. Feu una funció que indiqui si dues llistes són iguals.
2. Feu una funció que calculi el producte dels elements d'una llista.
3. Feu una funció que multiplica tots el nombres parells d'una llista d'enters.
4. Feu una funció que generi la llista de les k primeres potències de 2.
5. Feu una funció que calculi el producte escalar de dues llistes.

6. Feu una funció que donada una llista de llistes d'elements "l" i un element "n" ens torna la llista que indica quants cops apareix "n" en cada llista de "l".

7. Repetiu l'exercici de la sessió passada de generació dels nombres de Hamming utilitzant el `map`.

5 Definició de noves funcions d'ordre superior

En aquesta secció heu d'implementar una sèrie de noves funcions d'ordre superior.

1. Feu una funció `countIf` que donada una propietat i una llista, ens retorna el nombre d'elements de la llista que satisfan la propietat. Noteu que aquesta funció d'ordre superior existeix en llenguatges de tractament de fulls de càlcul com ara EXCEL.

2. Feu una funció `pam` que donada una llista d'enters i una llista de funcions d'enters a enters, ens torna la llista de llistes resultant d'aplicar cada una de les funcions de la segona llista als elements de la primera llista. Així, `pam [1,2,3] [(+1),(*2),(^2)]` retorna `[[2,3,4],[2,4,6],[1,4,9]]`

   Feu també la funció `pam2` on el resultat és la llista de llistes on cada llista és el resultat d'aplicar succesivament les funcions de la segona llista a cada element de la primera llista. Per tant, `pam2 [1,2,3] [(+1),(*2),(^2)]` retorna `[[2,2,1],[3,4,4],[4,6,9]]`

3. Feu la funció `filterfold` que fa el plegat dels elements que satisfan la propietat donada. Així, `filterfold even (*) 1 [4,7,2,4,9,3]` retorna 32.

4. Feu una funció `inserir` que donada una relació, un element i una llista ens retorna la llista amb l'element inserit segons la relació. Utilitzant aquesta funció aplicada només a la relació, és a dir, `(inserir f)` i el `foldr`, defineix una funció que donada una relació i una llista, ordeni la llista per inserció.