KS i Programmeringsparadigm 2016, del 1: Funktionell Programmering
2016-09-20 08.15-10.00 med efterföljande kamraträttning

Inga hjälpmedel är tillåtna. Skriv svaren direkt på blanketten. Bonuspoäng från Haskelllabbarna hösten 2016 kommer automatiskt att tillgodoräknas på denna KS. 12 poäng (utifrån max 20) krävs för godkänt. Sitt kvar ända till klockan 09.00. Då lämnar alla in samtidigt. Därefter tar kamraträttning vid.

1. (9 p)

   (a) Skriv en rekursiv funktion som heter `count`. Den ska räkna antal förekomster av ett givet element i en lista. Funktionen ska ta två parametrar där den första är elementet som söks och den andra är listan som funktionen letar i. Du får inte använda `filter`, `map`, `foldl`, `foldr` eller någon inbyggd `count`-funktion. Funktionen ska ha den angivna typsignaturen och får anropa hjälpfunktioner. (5 p).

   ```
   {- Returns the number of occurences of target in list. -}
   count::(Eq a)=>a->[a]->Int
   ```

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   .....

   (b) Ge 4 exempel på konkreta typer som list kan bestå av. (2 p)

   .....

   (c) Hur kan du återanvända funktionen `count` för att skapa en funktion `countSpaces` som räknar antalet mellanslag i en sträng? Du får högst använda 22 tecken kod. Vad kallas den egenskap hos Haskell som din funktionsdefinition måste använda? Du kan få full poäng på den här uppgiften även om du inte har implementerat `count`. Typsignaturen ingår inte i uppgiften. (2 p)

   .....

   .....

   .....

2. (5 p)

   (a) Studera följande datatyp:

```
{- A bank account with balance, name, accountNumber and passwordHash -}
data Account
    = Account { balance :: Integer
      , name            :: String
      , accountNumber   :: Integer
      , passwordHash    :: String
      }
    deriving (Show, Eq)
```

   Koden ovan sätter ihop befintliga datatyper till en ny typ. Vad kallas en sådan typ av typer? (2p)

   .....

   (b) Nedanstående rader evalueras i ghci. Beskriv övergripande (inte tecken för tecken utan med en kort beskrivande sammanfattning) vad ghci svarar för repektive rad och namnge den egenskap hos Haskell som det beror på (3p).

```
Prelude> [1,3..]
Prelude> let oddNumbers = [1,3..]
```

   .....

   .....

   .....

   .....

   .....

   .....

   .....

3. (6 p)

   (a) I ett termodynamiskt experiment skulle temperaturen hos en metall mätas noggrannt vid k tidpunkter under en nedkylning. Först beräknades de förväntade värdena och resultaten lagrades i listan `extrapolerat`, därefter genomfördes experimentet och resultaten lagrades i listan `uppmätt`. Båda listorna innehåller k tal av typen `Double`. Skriv en funktion som tar in de två listorna som parametrar och beräknar felkvadratsumman $\sum_{i=0}^{k-1}(u_i - e_i)^2$ där u_i är element i i listan `uppmätt`, e_i är element i i listan `extrapolerat` och k är längden på var och en av listorna. Alla index är nollindexerade. Din lösning ska använda minst en av `foldl`, `foldr` eller `map` och minst en funktion definierad med *lambdanotation*. Högre ordningens funktioner och lambda ska vara centrala för din lösning. Din funktion får endast anropa fördefinierade funktioner i

Haskells standardbibliotek. Du får med andra ord inte skriva egna hjälpfunktioner och inte heller anropa din egen funktion rekursivt. Kom ihåg typsignaturen.
Tips: För att slå ihop två listor, använd funktionen `zip` med typsignatur `zip :: [a] -> [b] -> [(a, b)]`. Den ger dig en lista med tupler av tal från samma index i respektive lista. (4p)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

(b) Ange typsignaturerna för de funktioner du använde i förra uppgiften (utom `zip` och även om du har använt någon av dem - lista inte heller typsignaturerna för `+`, `-`, `*` eller `/`.). (2p)

.....

.....

.....

.....

.....

.....

.....

.....