1. Übungsaufgabe zu
Fortgeschrittene funktionale Programmierung
Thema: Ströme, Generatoren und Selektoren
ausgegeben: Mi, 09.03.2016, fällig: Mi, 16.03.2016

Für dieses Aufgabenblatt sollen Sie Haskell-Rechenvorschriften zur Lösung der im folgenden angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens `AufgabeFFP1.hs` auf **oberstem Niveau in Ihrem Gruppenverzeichnis** ablegen. Kommentieren Sie Ihre Programme aussagekräftig und benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten.

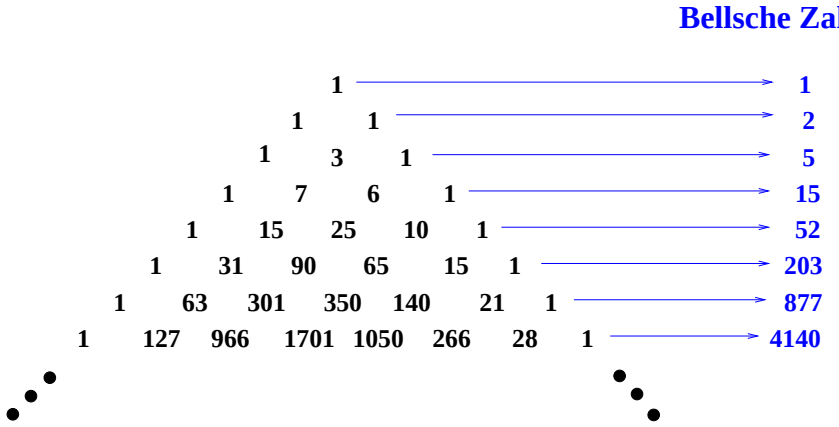
- Schreiben Sie eine 0-stellige Haskell-Rechenvorschrift `powersOfThrees :: [Integer]`, die den Strom der 3er-Potenzen liefert, d.h. den Strom `[1,3,9,27,81,243,...`.

  Versuchen Sie, `powersOfThrees` rekursiv ausschließlich mit Hilfe von `map` (sowie dem Listenkonstruktor ":" und arithmetischen Funktionen zu definieren).

- Die *Stirlingschen Zahlen* $S(n,k)$ geben die Anzahl der Zerlegungen einer n-elementigen Menge in k paarweise verschiedene nichtleere Teilmengen an, deren Vereinigung die Gesamtmenge ergibt. Für die Menge $\{a,b,c\}$ gilt z.B.:

  $$\begin{array}{lcll} S(3,1) & = & 1 & // \ \{\,\{a,b,c\}\,\} \\ S(3,2) & = & 3 & // \ \{\,\{\{a,b\},\{c\}\},\{\{a\},\{b,c\}\},\{\{a,c\},\{b\}\}\,\} \\ S(3,3) & = & 1 & // \ \{\,\{a\},\{b\},\{c\}\,\} \end{array}$$

  Ähnlich zur Anordnung der Binomialkoeffizienten im Pascalschen Dreieck lassen sich auch die Stirlingschen Zahlen bequem in einem Dreieck anordnen.



Stirlingsches Dreieck

  Eine Zahl im Stirlingschen Dreieck wird dabei nach folgender (informeller) Rekursionsformel berechnet:

  – Eine Zahl im Rumpf des Dreiecks ergibt sich aus den beiden unmittelbar darüber stehenden Zahlen, indem zur ersten Zahl das Produkt aus der zweiten Zahl und der Position in der Reihe der gesuchten Zahl addiert wird.

Beispielsweise ergibt sich die Zahl 10 in der 5-ten Reihe des Dreiecks als Wert des Ausdrucks $6 + 4 * 1$ (6 und 1 sind die über der 10 stehenden Zahlen, 10 selbst steht an der 4-ten Position in seiner Zeile).

Formal sieht die Rekursionsformel folgendermaßen aus:

$$S(n+1,k) = S(n,k-1) + k * S(n,k) \quad \text{für } n = 1,2,3,\ldots$$

Wir bezeichnen die Zeilen dieses Stirlingschen Dreiecks als *Stirling-Tupel*.

Schreiben Sie eine 0-stellige Haskell-Rechenvorschrift `sd :: [[Integer]]`, die den Strom der Stirling-Tupel liefert, d.h. den Strom `[[1],[1,1],[1,3,1],[1,7,6,1],...`

Versuchen Sie, `sd` rekursiv ausschließlich mit Hilfe von `zipWith` (sowie den Standardkonstruktoren und -operatoren auf Listen und arithmetischen Funktionen) zu definieren.

Der Generator/Selektor-Aufruf soll beispielsweise folgendes Resultat liefern:

```
take 5 sd ->> [[1],[1,1],[1,3,1],[1,7,6,1],[1,15,25,10,1]]
```

- Addiert man die Zeilen im Stirlingschen Dreieck auf, so erhält man die *Bellschen Zahlen*. Sie geben an, auf wieviele Möglichkeiten eine Menge in paarweise verschiedene nichtleere Teilmengen zerlegt werden kann.

  Schreiben Sie eine 0-stellige Haskell-Rechenvorschrift `bn :: [Integer]`, die den Strom der Bellschen Zahlen aus dem Strom der Stirling-Tupel berechnet, d.h. den Strom `[1,2,5,15,52,...`

  Der Generator/Selektor-Aufruf soll beispielsweise folgendes Resultat liefern:

  ```
  take 5 bn ->> [1,2,5,15,52]
  ```

- Seien $m, n, p \in \mathbb{N}$ mit $m, n, p > 0$. Ein Tripel (m, n, p) heißt *Pythagoreisches Tripel*, wenn gilt: $m \leq n \leq p$ und $m^2 + n^2 = p^2$.

  Schreiben Sie eine 0-stellige Haskell-Rechenvorschrift

  ```
  pt :: [(Integer,Integer,Integer)]
  ```

  die den Strom der Pythagoreischen Tripel liefert, wobei die Tripel in der folgenden Weise angeordnet sein sollen: Die Komponenten jedes Tripels sind aufsteigend geordnet; ein Tripel t_1 kommt im Strom früher als ein Tripel $t2$ vor, wenn die dritte Komponente von t_1 kleiner als die dritte Komponente von $t2$ ist, d.h. `pt` entspricht dem Strom `[(3,4,5),(6,8,10),(5,12,13),(9,12,15),...`

Wichtiger Hinweis: Die Aufgaben werden für die automatische Überprüfung unter Hugs auf der Maschine g0.complang.tuwien.ac.at ausgeführt. Wenn Sie für die Programmentwicklung eine andere Maschine oder einen anderen Interpretierer benutzen, überprüfen Sie auf jeden Fall rechtzeitig vor dem Abgabezeitpunkt, dass Ihre Programme auch unter Hugs auf der Maschine g0.complang.tuwien.ac.at wie von Ihnen beabsichtigt laufen.