

1. Übungsaufgabe zu
Fortgeschrittene funktionale Programmierung
Thema: Ströme, Generatoren und Selektoren
ausgegeben: Di, 12.03.2013, fällig: Di, 19.03.2013

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 `pow2s :: [Integer]`, die den Strom der 2er-Potenzen liefert, d.h. den Strom `[1,2,4,8,16,32,...]`.
Versuchen Sie, `pow2s` rekursiv ausschließlich mit Hilfe von `map` (sowie dem Listenkonstruktor `“:”` und arithmetischen Funktionen zu definieren).
- Im folgenden ist ein Anfangsstück des Pascalschen Dreiecks dargestellt:

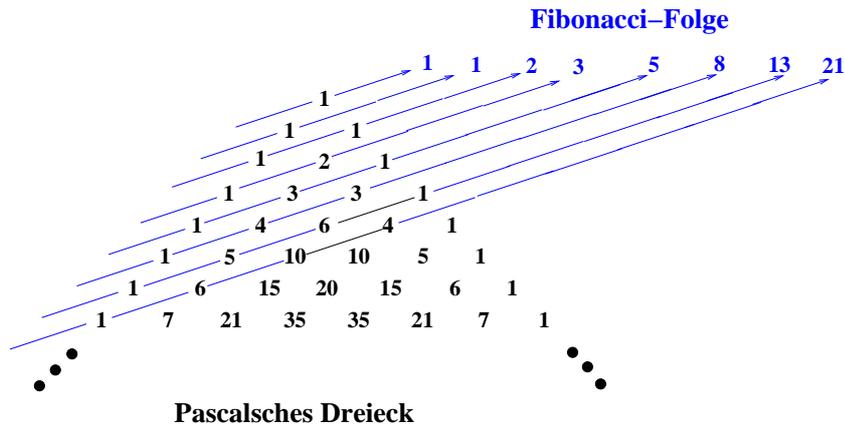
```
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
...
```

Wir bezeichnen die Zeilen dieses Dreiecks als *Pascaltupel*.

Schreiben Sie eine 0-stellige Haskell-Rechenvorschrift `pd :: [[Integer]]`, die den Strom der Pascaltupel liefert, d.h. den Strom `[[1],[1,1],[1,2,1],[1,3,3,1],...]`.

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

- Die folgende Abbildung veranschaulicht, dass das Pascalsche Dreieck implizit die Folge der Fibonacci-Zahlen enthält.



Schreiben Sie eine Haskell-Rechenvorschrift `fibdiag`

```
fibdiag :: Integer -> [Integer]
```

vor, die angewendet auf ein positives Argument n , $n > 0$, die Liste der in Pfeilrichtung angeordneten Summanden liefert, die sich zur n -ten Fibonacci-Zahl summieren.

Folgende Beispiele zeigen das Aufruf/Resultatverhalten von `fibdiag`:

```
fibdiag 1 ->> [1]
fibdiag 2 ->> [1]
fibdiag 3 ->> [1,1]
fibdiag 5 ->> [1,3,1]
fibdiag 8 ->> [1,6,10,4]
```

- Schreiben Sie mithilfe von `fibdiag` eine Haskell-Rechenvorschrift `fibdiags`, die den Strom der Diagonalen aus der obigen Abbildung liefert:

```
fibdiags :: [[Integer]]
```

- Geben Sie weiters eine Haskell-Rechenvorschrift `fibspd` an, die den Strom der Fibonacci-Zahlen in der durch die Abbildung nahegelegten Weise aus dem Pascalschen Dreieck generiert.

```
fibspd :: [Integer]
```

Die Implementierung des Stroms `fibspd` soll sich dabei geeignet auf den Strom `pd` (oder auf `fibdiags`) aus den vorherigen Teilaufgaben abstützen.

Die Generator/Selektor-Aufrufe sollen folgende Resultate liefern:

```
take 5 pd ->> [ [1], [1,1], [1,2,1], [1,3,3,1], [1,4,6,4,1] ]
take 5 fibdiags ->> [ [1], [1], [1,1], [1,2], [1,3,1] ]
take 5 fibspd ->> [1,1,2,3,5]
```