1. Übungsaufgabe zu
Fortgeschrittene Funktionale Programmierung
Themen: Ströme, Generator/Selektor-Prinzip
ausgegeben: 15.05.2006, fällig: 24.05.2006

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` in Ihrem Homeverzeichnis ablegen, wie gewohnt auf oberstem Niveau. Kommentieren Sie Ihre Programme aussagekräftig und benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten.

- Schreiben Sie eine Haskell-Rechenvorschrift `facsFrom :: Int -> [Int]`, die den Strom der "Fakultätszahlen" ausgibt beginnend mit der durch das Argument von `facsFrom` vorgegebenen Fakultätszahl. Der Aufruf `facsFrom 1` soll also die Ausgabe `[1,1,2,6,24,120,720,...` liefern, der Aufruf von `facsFrom 5` die Ausgabe `[24,120,720,...`. Für nicht-positive Argumente sei das Resultat der Anwendung von `facsFrom` der nur aus Nullen bestehende Strom.

- Schreiben Sie unter Benutzung der Rechenvorschrift `facsFrom` aus dem vorigen Aufgabenteil und des Generator/Selektor-Prinzips Haskell-Rechenvorschriften
  - `fac :: Int -> Int`, die angewendet auf $n \geq 0$, $n \in \mathbb{N}$, die Ausgabe $n!$ liefert,
  - `facN :: Int -> [Int]`, die angewendet auf $n \geq 0$, $n \in \mathbb{N}$, die Liste der ersten n Fakultätszahlen liefert, beispielsweise soll der Aufruf `facN 4` die Ausgabe `[1,1,2,6]` liefern,
  - `facVB :: Int -> Int -> [Int]`, die angewendet auf $b \geq v \geq 0$, $b, v \in \mathbb{N}$, v erstes, b zweites Argument von `facVB` die Liste der v-ten bis b-ten Fakultätszahlen liefert, beispielsweise soll der Aufruf `facVB 2 4` die Ausgabe `[1,2,6]` liefern.

- Schreiben Sie eine Haskell-Rechenvorschrift `fsum :: Int -> Int -> [Int]`, die angewendet auf die Argumente m und n als Resultat den Strom liefert, der sich aus der komponentenweisen Addition der Ströme ergibt, die aus den Aufrufen `facsFrom m` und `facsFrom n` resultieren.

- Schreiben Sie analog zu den Funktionen `facN` und `facVB` auch Funktionen `fsumN` und `fsumVB`.

- Natürliche Zahlen, in deren Primfaktorzerlegung nur die Primfaktoren 2, 3 und 5 (null- oder auch mehrmals) vorkommen, heißen *Hammingzahlen*, d.h. die Menge der natürlichen Zahlen der Form $2^i * 3^j * 5^k$, $i, j, k \geq 0$. Schreiben Sie eine Haskell-Rechenvorschrift `hamming :: [Int]`, die den Strom der Hamming-Zahlen in aufsteigender Folge als Ausgabe liefert. Der Aufruf von `hamming` soll also die Ausgabe `[1,2,3,4,5,6,8,9,10,12,15,...` liefern.

- Schreiben Sie analog zu den Funktionen `facN` und `facVB` auch Funktionen `hamN` und `hamVB`.

- Schreiben Sie eine Haskell-Rechenvorschrift `sumPrimes :: Int -> Int`, die angewendet auf Argumente n größer oder gleich 0 die Summe der ersten n Primzahlen liefert. Die kleinste Primzahl ist 2; 1 ist keine Primzahl. Der Aufruf `sumPrimes 5` soll also die Ausgabe 28 liefern. Nutzen Sie zur Definition von `sumPrimes` wieder das Generator/Selektor-Prinzip aus.

- Der Wert der Sinusfunktion sin an der Stelle x wird durch die Reihe

$$sin\ x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + \ldots$$

  approximiert. Schreiben Sie eine Haskell-Rechenvorschrift `sins :: Float -> [Float]`, die den Strom der k-Präfixe der Approximation von sin angewendet auf x liefert. Die Ausgabe von `sins` soll also der Strom $[x, x - \frac{x^3}{3!}, x - \frac{x^3}{3!} + \frac{x^5}{5!}, \ldots$ sein.

- Sehen Sie auch eine Funktion `sinskPref :: Int -> Float -> Float` vor, die angewendet auf n, $n \in \mathbb{N}_0$, und x den n-Präfix von `sin x` liefert, für $n = 0$ also x, für $n = 2$ also $x - \frac{x^3}{3!} + \frac{x^5}{5!}$.