Institut für Informatik **Ludwig-Maximilians-Universität München**
Prof. Dr. François Bry

Programmierung und Modellierung, SoSe 16
Übungsblatt 7

Abgabe: bis Mo 30.05.2016 10:00 Uhr

Besprechung: ab Di 31.05.2016

In der 7. Semesterwoche (23.05.-27.05.) finden *keine* Übungen statt.

Aufgabe 7-1 Rekursion mit Listen

Implementieren Sie die folgenden Funktionen in Haskell. Es gelten die gleichen Spielregeln wie bei Aufgabe 4-1: Keine Funktionen der Standardbibliothek, mit Ausnahme von Listenoperatoren und einfachen arithmetischen Funktionen.

a) Implementieren Sie eine Funktion, die das kleinste Element einer übergebenen Liste ausgibt.

b) Implementieren Sie eine Funktion, die eine Zahl an der richtigen Stellen einer sortierten Liste einordnet.

c) Implementieren Sie eine Funktion, welche das erste Vorkommen einer als Parameter übergebenen Zahl aus einer Liste entfernt.

d) Definieren Sie eine Funktion, welche eine Liste vom Typ `[Int]` sortiert.

Hinweis: Funktionen aus den vorherigen Teilaufgaben könnten sich hier als nützlich erweisen.

Aufgabe 7-2 Primfaktorzerlegung

Jede Zahl kann in Primfaktoren zerlegt werden. Die Primfaktoren einer natürlichen Zahl n sind die Primzahlen, die miteinander multipliziert wieder n ergeben. Implementieren Sie eine Funktion, welche die Primfaktorzerlegung einer natürlichen Zahl bestimmt.

Beispiel: Von `12` ist die Primfaktorzerlegung `[2, 2, 3]`, da `12 = 2 * 2 * 3`.

Hinweis: Eines der Ergebnisse aus Aufgabe 4-1 und das Sieb des Eratosthenes aus Aufgabe 4-2 könnten sich hier als nützlich erweisen.

Aufgabe 7-3 Arithmetische Operationen ohne die Standardbibliothek

Gegeben sind die folgenden Definitionen `succ'` und `pred'`, die jeweils den direkten Nachfolger, bzw. direkten Vorgänger eines ganzzahligen Werts bestimmen.

```
succ' :: Integer -> Integer
succ' x = x + 1

pred' :: Integer -> Integer
pred' x = x - 1
```

Die beiden Funktionen finden Sie auch im Verzeichnis `u07/7-3/` in der Datei `angabeSuccPred.hs`.

Implementieren Sie ausschließlich unter Benutzung der gegebenen Funktionen die folgenden Funktionen.

a) Implementieren Sie eine Funktion `plus :: Integer -> Integer -> Integer`, die zwei `Integer`-Werte addiert, sowie eine Funktion `minus :: Integer -> Integer -> Integer`, die den zweiten Wert vom ersten abzieht. Benutzen Sie dabei keine Funktionen der Standardbibliothek.

b) Implementieren Sie jetzt aufbauend auf Ihrem Ergebnis von Teilaufgabe a) eine Funktion `mult :: Integer -> Integer`, die zwei `Integer`-Werte miteinander multipliziert.

c) Implementieren Sie jetzt aufbauend auf den Ergebnissen der vorherigen Teilaufgaben einen Funktion `mod :: Integer -> Integer`, welche den Rest bei der Ganzzahldivison der ersten durch die zweiten Zahl zurückgibt.

Aufgabe 7-4 Abstiegsfunktion

Gegeben sei die folgende Implementierung einer Funktion, welche die Länge einer übergebenen Liste bestimmt.

```
length' []     = 0
length' (x:xs) = 1 + length' xs
```

Die Funktion finden Sie auch im Verzeichnis `u07/7-4/` in der Datei `angabeLaenge.hs`.

Zeigen Sie mit Hilfe einer Abstiegsfunktion, dass die Funktion `length'` terminiert.