Praktische Informatik in der Wirtschaft
Westfälische Wilhelms-Universität Münster
Prof. Dr. Herbert Kuchen 9. Januar 2017

---

10. Übung zu Informatik I im WS 16/17
Abgabe: 16. Januar 2017, 12 $\frac{00}{}$ Uhr.

Wichtig: Sämtliche Haskell-Abgaben müssen in der ersten Zeile eine Moduldeklaration führen. Diese sieht beispielsweise so aus:

```
module Modulname where
```

Der zu verwendende `Modulname` steht in der jeweiligen Aufgabe. Beachten Sie außerdem: Die Fehlermeldung **"Could not compile all files"** kann mit einem falschen oder fehlenden Modulnamen oder aber mit einer falschen Funktionsdefinition zu tun haben!

Aufgabe 23 - Gruppenabgabe (4 + 4 + 4 = 12 Punkte)

Geben Sie für die folgenden Ausdrücke die Reduktionsfolge gemäß der am weitesten linkst stehenden, innersten Reduktion (leftmost innermost reduction) und der am weitesten linkst stehenden, äußersten Reduktion (leftmost outermost reduction) an!

a)
```
sqr :: Int -> Int
sqr x = x*x

second :: (a, b) -> b
second (_, x) = x
```

Ausdruck: `second (sqr 2, sqr 4)`

b)
```
ones :: [Int]
ones = 1:ones

head :: [a] -> a
head (x: l) = x
```

Ausdruck: `head ones`

c)
```
map :: (a -> b) -> [a] -> [b]
map _ []     = []
map f (x:xs) = f x : map f xs

inc :: Int -> Int
inc x  = x + 1
```

Ausdruck: `map inc [2, 3]`
Hinweis: `[2, 3]` ist eine verkürzte Schreibweise für `2 : (3 : [])`

Bitte laden Sie Ihre Lösung als Pdf-Datei hoch!

Hinweis: Bei der leftmost outermost reduction wird der am weitesten linkst stehende, äußerste reduzierbare Teilausdruck (Redex) zuerst aufgelöst. Bei der leftmost innermost reduction wird der am weitesten linkst stehende, innerste Redex zuerst aufgelöst. Nehmen wir folgendes Beispiel: `sqr (sqr 2)`

Lefmost outermost:

```
sqr (sqr 2)
(sqr 2) * (sqr 2)
(2 * 2) * (sqr 2)
4 * (sqr 2)
4 * (2 * 2)
4 * 4
16
```

Lefmost innermost:

```
sqr (sqr 2)
sqr (2*2)
sqr 4
4 * 4
16
.
.
```

Aufgabe 24 - Einzelabgabe (8 Punkte)

Implementieren Sie die Haskell-Funktionen `verzahnen` innerhalb eines Moduls `Verzahnen`, welche zwei Listen vom selben Typ erwartet und ein Liste von diesem Typ zurückgibt! Dabei soll die zurückgegebene Liste alle Elemente beider Listen enthalten, wobei immer abwechselnd ein Element der ersten und anschließend von der zweiten Liste enthalten sein soll. Sobald eine der beiden Listen leer ist, werden nur noch Elemente der anderen Liste angehängt. Nachfolgend sind zwei Beispiele zu sehen, welche dieses Verfahren verdeutlichen sollen.

$$\underset{\text{a}}{\begin{bmatrix}1\\2\end{bmatrix}} \quad \underset{\text{b}}{\begin{bmatrix}3\\4\\7\end{bmatrix}} \quad \rightsquigarrow \quad \underset{\text{Ergebnis}}{\begin{bmatrix}1\\3\\2\\4\\7\end{bmatrix}} \qquad \underset{\text{a}}{\begin{bmatrix}1\\2\\6\\8\end{bmatrix}} \quad \underset{\text{b}}{\begin{bmatrix}3\end{bmatrix}} \quad \rightsquigarrow \quad \underset{\text{Ergebnis}}{\begin{bmatrix}1\\3\\2\\6\\8\end{bmatrix}}$$

Aufgabe 25 - Gruppenangabe (6 + 4 = 10 Punkte)

a) Implementieren Sie eine Haskell-Funktion `zahlenFolge`, welche eine Liste von Int zurückgibt, wobei jedes Element dieser Liste durch eine der Zahlen 5, 7 oder 9 teilbar sein muss. Die Liste sollte dabei alle möglichen positiven Zahl in aufsteigender Reihenfolge enthalten und unendlich sein. Implementieren Sie Ihre Funktion also so, dass die Liste lazy erstellt werden kann und beispielsweise folgender Befehl zu keinem Fehler führt: `take 10 zahlenFolge`. Verwenden Sie als Modul-Bezeichnung bitte `ZahlenFolge`.

Beispiel:

```
take 10 zahlenFolge
[5, 7, 9, 10, 14, 15, 18, 20, 21, 25]
```

b) Erweitern Sie Ihre Funktion `zahlenFolge` so, dass Sie eine Liste von Int entgegennimmt. Die zurückgegebene Liste folgt den selben Bedingungen wie in a) gefordert. Als Teiler werden nun aber anstelle von 5, 7 und 9 alle Zahlen der übergebenen Liste verwendet.

Beispiel:

```
take 10 (zahlenFolge [2, 5])
[2, 4, 5, 6, 8, 10, 12, 14, 15, 16]
```

Hinweis: Sie können bei Ihrer Implementierung auf die Modulo Funktion von Haskell zurückgreifen:

```
mod :: Int -> Int -> Int
```

Wobei zum Beispiel:

```
mod 5 3 ⇝ 2
```

Anmerkung: *Versehen Sie ihre Abgaben bitte mit **Namen**, **Matrikelnummern**, **E-Mail-Adressen** und **Studiengängen** der beteiligten Bearbeiter/innen und laden Sie diese in der entsprechenden Aktivität im Learnweb hoch. Aufgaben, die mit dem Hinweis **Gruppenabgabe** versehen sind, dürfen mit maximal 3 Bearbeiter/innen gelöst werden. Aufgaben mit dem Hinweis **Einzelabgabe** müssen von jedem Studenten und jeder Studentin eigenständig gelöst und abgegeben werden. Viele der Aufgaben werden über das **EASy** (E-Assessment System) System im Learnweb eingereicht und automatisch vorausgewertet. Um Probleme bei der Abgabe wegen Überlastung des EASy-Servers zu vermeiden, würde ich Sie bitten, eine Abgabe auf den letzten Drücker nach Möglichkeit zu vermeiden. Bei Fragen zum Übungsbetrieb wenden Sie sich bitte an ihren/ihre Tutor/in oder an Tobias Reischmann.*