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

**Programmierung und Modellierung**, SoSe 16
**Übungsblatt 4**

*Abgabe: bis Mo 02.05.2016 10:00 Uhr*

Besprechung: ab Di 03.05.2016

## Aufgabe 4-1 Rekursion mit Listen

Implementieren Sie die folgenden Funktionen „von Hand“, d.h. ohne auf Funktionen der Standardbibliothek von Haskell zurückzugreifen. Einfache Listenoperatoren wie : und ++ sowie arithmetische Funktionen (+, −,..) sind erlaubt und müssen nicht selbst implementiert werden.

Geben Sie (soweit möglich) je eine endrekursive und eine nicht endrekursive Implementierung der gesuchten Funktion an und vergleichen Sie die Laufzeiten und den Speicherbedarf der beiden Implementierungen mit dem Haskell-Package Criterion (siehe Vorlesung).

a) Implementieren Sie eine Funktion, welche die Länge einer Liste bestimmt.

b) Implementieren Sie eine Funktion, die ein Element und eine Liste übergeben bekommen und überprüft, ob das Element in der Liste enthalten ist.

c) Implementieren Sie eine Funktion, die eine übergebene Liste umdreht. So soll beispielsweise aus der Liste `[1, 2, 3]` die Liste `[3, 2, 1]` werden.

d) Implementieren Sie eine Funktion die zwei übergebene Listen aneinanderhängt. So soll beispielsweise aus den Listen `[1, 2, 3]` und `[4, 5, 6]` die Liste `[1, 2, 3, 4, 5, 6]` werden.

## Aufgabe 4-2 Sieb des Eratosthenes

Das Sieb des Eratosthenes ist ein Algorithmus um Primzahlen bis zu einer bestimmten Grenze zu bestimmen. Die Idee dahinter ist schnell erklärt: Ausgehend von einer Liste, welche die Zahlen von 2 bis zur gewünschten Grenze enthält werden von vorne Zahlen eingesammelt. Immer wenn eine Zahl eingesammelt wurde, werden alle Vielfachen von ihr aus der Liste entfernt. Die Liste der eingesammelten Zahlen sind die Primzahlen bis zur gewählten Grenze.

Beispiel für Liste von 2 bis 15

[**2**, 3, ~~4~~, 5, ~~6~~, 7, ~~8~~, 9, ~~10~~, 11, ~~12~~, 13, ~~14~~, 15]
[2, **3**, ~~4~~, 5, ~~6~~, 7, ~~8~~, ~~9~~, ~~10~~, 11, ~~12~~, 13, ~~14~~, ~~15~~]
[2, 3, ~~4~~, **5**, ~~6~~, 7, ~~8~~, ~~9~~, ~~10~~, 11, ~~12~~, 13, ~~14~~, ~~15~~]
[2, 3, ~~4~~, 5, ~~6~~, **7**, ~~8~~, ~~9~~, ~~10~~, 11, ~~12~~, 13, ~~14~~, ~~15~~]
[2, 3, ~~4~~, 5, ~~6~~, 7, ~~8~~, ~~9~~, ~~10~~, **11**, ~~12~~, 13, ~~14~~, ~~15~~]
[2, 3, ~~4~~, 5, ~~6~~, 7, ~~8~~, ~~9~~, ~~10~~, 11, ~~12~~, **13**, ~~14~~, ~~15~~]

Implementieren Sie das Sieb des Eratosthenes in Haskell. Benutzen Sie dabei eine *List Comprehension*.

### Aufgabe 4-3 Terminierungsbeweis für Fibonacci-Zahlen

Beweisen Sie, dass die folgende Implementierung für die rekursive Berechnung der Fibonacci-Zahlen für Eingabewerte größer oder gleich 0 terminiert.

```
fib 0 = 1
fib 1 = 1
fib n = fib (n-1) + fib (n-2)
```