WiSe 2015/2016

# Funktionale Programmierung

**2. Übungsblatt** (Abgabe: Mo., den 02.11. um 10:10 Uhr)

Prof. Dr. Margarita Esponda

---

**Ziel:** Auseinandersetzung mit Listen und einfachen rekursiven Funktionsdefinitionen.

**1. Aufgabe** (2 Punkte)

Betrachten Sie folgende Haskell-Funktionsdefinition, die überprüfen soll, ob die eingegebene Zahl **n** eine ungerade Zahl ist.

```
ungerade :: Integer -> Bool
ungerade n = rem n 2 == 1
```

a) Warum ist diese Funktionsdefinition inkorrekt?

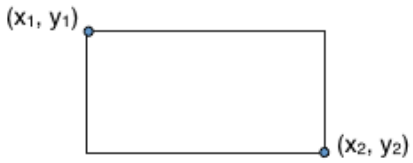
b) Geben Sie eine korrekte Funktionsdefinition.

**2. Aufgabe** (8 Punkte)

Nehmen Sie an, wir haben folgende Datentyp-Synonyme definiert:

```
type Point  = (Double, Double)
type Rectangle = (Point, Point)
```

Ein Rechteck des Datentyps "Rectangle" ist immer parallel zum Koordinatensystem und wird durch die Koordinaten der Ecken oben links und unten rechts $((x_1,y_1),(x_2,y_2))$ definiert.




a) Definiere unter Verwendung des **Rectangle**-Datentyps folgende Funktionen:

```
area :: Rectangle -> Double
contains :: Rectangle -> Rectangle -> Bool   -- Testet, ob eines der Rechtecke das
                                                andere Rechteck beinhaltet
overlaps :: Rectangle -> Rectangle -> Bool   -- Testet, ob die Rechtecke sich überlappen
```

b) Testen Sie Ihre Funktionen mindestens mit folgenden Beispielen:

Anwendungsbeispiele:

```
contains ((2,8),(4,5)) ((1,7),(3,4))       =>  False
contains ((2,12),(10,2)) ((3,8),(7,5))     =>  True

overlaps ((3,9),(7,6)) ((6,7),(10,3))      => True
overlaps ((3,9),(9,3)) ((1,12),(12,1))     => True
overlaps ((3,9),(9,3)) ((12,12),(12,1))    => False
```

**3. Aufgabe** (3 Punkte)

Betrachten Sie folgende Funktionsdefinition.

```
collList :: Integer -> [Integer]
collList    1     =  [1]
collList  (n+1)  =  (n+1): collList (next (n+1))
                    where
                        next  n  | mod n 2 == 0  =  div n 2
                                 | otherwise      =  3*n + 1
```

Reduzieren Sie den folgenden Ausdruck, indem Sie alle einzelnen Schritte bis zur Normalform aufschreiben.

```
collList  5
```

**4. Aufgabe** (4 Punkte)

Definieren Sie eine Haskell-Funktion **sumDigits**, die die Quersumme einer Zahl so lange berechnet, bis das Ergebnis nur aus einer Ziffer besteht (Zahl zwischen 0-9).

Anwendungsbeispiel: **sumDigits** 1245091 => 4

**5. Aufgabe** (4 Punkte)

Schreiben Sie eine rekursive Funktion, die bei Eingabe einer positiven ganzen Zahl **n** alle Zahlen zwischen **1** und **n** aufsummiert, die ungerade und durch 3 oder 11 teilbar sind.

**6. Aufgabe** (8 Punkte)

Innerhalb eines Rechners werden Wahrheitswerte nur mit Hilfe der Binärzahlen 1 und 0 dargestellt. Nehmen wir an, wir haben folgende Haskell-Funktionen definiert:

```
true :: Int
true = 1

false :: Int
false = 0
```

a) Definieren Sie folgende eigene logische Funktionen **negation**, **und**, **oder** und **exoder** nur mit Hilfe der arithmetischen Operationen -, + und *.

Der Hamming-Abstand zwischen zwei Binärzahlen mit gleicher Länge ist gleich der Anzahl der unterschiedlichen Bitstellen.

b) Verwenden Sie Ihre **exoder** Funktion, um eine rekursive Funktion zu definieren, die bei Eingabe zweier Binärzahlen den Hamming-Abstand berechnet.
Verwenden Sie die **error**-Funktion für den Fall, dass die Bit-Listen nicht gleich lang sind.

Anwendungsbeispiel:

**hamming_distance** [1,0,0,1,1,0,0,1] [1,0,0,1,0,0,1,1] => 2

**7. Aufgabe** (4 Punkte)
Programmieren Sie eine rekursive Funktion, die an einer bestimmten Position ein neues Element in eine Liste einfügt.

Verwenden Sie die **error**-Funktion für den Fall, dass die Position großer als die Länge der Liste ist.

Anwendungsbeispiel:

**insertElem** `a' 3 ['m','m','m','m','m','m'] => "mmmammm"

**Wichtige Hinweise:**

1) Verwenden Sie geeignete Namen für Ihre Variablen und Funktionsnamen, die den semantischen Inhalt der Variablen oder die Semantik der Funktionen wiedergeben.

2) Verwenden Sie vorgegebene Funktionsnamen, falls diese angegeben werden.

3) Kommentieren Sie Ihr Programme.

4) Verwenden Sie geeignete lokale Funktionen und Hilfsfunktionen in Ihren Funktionsdefinitionen.

5) Schreiben Sie in allen Funktionen die entsprechende Signatur.