6. Übung **Informatik A** **WS 13/14**

Klaus Kriegel **Abgabe: 02.12.2013, 12:00 Uhr**

Allgemeine Anforderungen bei Programmieraufgaben:
Die Haskell–Dateien mit dem Code müssen per email an den Tutor geschickt werden. Zusätzlich sollte der Code gedruckt oder handschriftlich abgegeben werden. Alle Lösungen müssen die folgenden Forderungen erfüllen:

- Auswahl sinnvoller Bezeichner für alle Funktionen;
  Bereits in der Aufgabenstellung spezifizierte Funktionsnamen sind zu übernehmen.
- Für alle neu definierten Funktionen ist die Signatur anzugeben.
- Die Programme müssen nicht nur lauffähig, sondern auch mit verschiedenen Eingabebeispielen getestet sein.
- Ausreichende Kommentierung der Lösungen;

Aufgabe 0: **Hugs** (0 Punkte)

Machen Sie sich mit Hugs vertraut! Implementieren Sie einige der Beispiele, die in Vorlesung und Übung besprochen wurden, bauen Sie absichtliche Fehler ein (Funktionsname mit Großbuchstaben am Anfang, Wächtersymbol nicht eingerückt, Parameterliste widersprüchlich zur Signatur, ...) und sehen Sie sich die Fehlermeldungen an.

Aufgabe 1: **Warming up - Fallunterscheidungen** (8 Punkte)

Implementieren Sie die folgenden Funktionen in Haskell. Verwenden Sie keine anderen als die angegebenen Datentypen und keine vordefinierten Funktionen wie `min, max, abs`. Sie können aber selbst geeignete Hilfsfunktionen definieren.

a) Die Funktion `avgIncluded :: Int -> Int -> Int -> Bool` erhält drei beliebige `Int`–Werte als Eingabe und soll `True` ausgeben, wenn einer der Werte der Durchschnitt dieser drei Werte ist, sonst `False`.

b) Die Funktion `smallestDifference3` erhält drei beliebige `Int`–Werte als Eingabe und soll den kleinsten Absolutbetrag einer Differenz aus zwei der drei Eingabewerte ausgeben.

c) Die Funktion `smallestDifference4` soll das analoge Problem wie in b) für vier Eingabewerte lösen. Versuchen Sie dazu, die Funktion aus b) zu verwenden.

d) Die Funktion `antival4` erhält vier beliebige `Bool`–Werte und soll die Antivalenz aus diesen Werten bestimmen.

Aufgabe 2: **Warming up - Rekursion** (8 Punkte)

Implementieren Sie die folgenden Funktionen in Haskell.

a) Die Funktion `maxExp2 :: Int -> Int` soll für ein gegebenes $n \in \mathbb{Z}$ das größte Exponent $k \in \mathbb{N}$ berechnen, so dass die Zweierpotenz 2^k ein Teiler von n ist. Für $n = 0$ soll $k = 0$ ausgegeben werden.

b) Die Funktion `minPow3 :: Int -> Int` soll für ein gegebenes $n \in \mathbb{N}$ die kleinste Dreierpotenz 3^k ($k \in \mathbb{N}$) berechnen, so dass $n \leq 3^k$.

c) Die Funktion `thirdRoot :: Int -> Int` soll für ein gegebenes $n \in \mathbb{N}$ die abgerundete dritte Wurzel, also das gößte $k \in \mathbb{N}$ mit $k^3 \leq n$ berechnen.

d) Die Funktion `sumOfOdds :: Int -> Int` soll für ein gegebenes $n \in \mathbb{N}$ die Summe aller ungeraden Zahlen k mit $1 \leq k \leq n$ berechnen.

Aufgabe 3: **Pseudocode verstehen** (6 Punkte)

Betrachten Sie die folgenden Prozeduren und erkennen Sie, was dabei eigentlich berechnet wird. Beschreiben Sie das Ergebnis mit einer kurzen Formel oder mit einem einfachen Satz und begründen Sie die Antwort.

```
magic1(n):  n∈ ℝ⁺
k=1

while (n > 0)
      k = 3-k
      n = n-1
return k
```

```
magic2(x):  x∈ ℚ⁺
j=1
k=0
while (j < x)
      j = 2*j
      k = k+1
return k
```

```
magic3(n,m):  n,m∈ ℕ
j=min(n,m)
k=max(n,m)
p = 0
if (k -j <= 1) then p = k
    else p = magic3(j+1,k-1)
return p
```