2. Aufgabenblatt zu Funktionale Programmierung vom 20.10.2004, fällig: 27.10.2004

Themen: *Funktionen auf ganzen Zahlen und Listen*

Für dieses Aufgabenblatt sollen Sie Haskell-Rechenvorschriften zur Lösung der unten angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens `Aufgabe2.lhs` ablegen. Sie sollen also ein "literate Script" schreiben. Versehen Sie wie auf dem ersten Aufgabenblatt auch hier alle Funktionen, die Sie zur Lösung brauchen, mit ihren Signaturen und kommentieren Sie Ihre Programme wieder aussagekräftig. Benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten. Im einzelnen sollen Sie folgende Problemstellungen bearbeiten:

1. Seien m und n zwei natürliche Zahlen mit $m, n \geq 1$. Eine natürliche Zahl kgv heißt *kleinstes gemeinsames Vielfaches* von m und n, falls es zwei natürliche Zahlen p und q, $p, q \geq 1$, mit $m*q = n*p = kgv$ gibt und für jede andere natürliche Zahl gv mit dieser Eigenschaft gilt: $kgv \leq gv$. Schreiben Sie eine Rechenvorschrift `kgv`, die zwei ganzzahlige Argumente erwartet (einzeln, also kein Paar ganzzahliger Argumente!). Sind beide Argumente größer oder gleich 1, so liefert die Funktion `kgv` als Ergebnis das kleinste gemeinsame Vielfache der beiden Argumente, ansonsten 0. Verwenden Sie in der Signatur der Funktion `kgv` ausschließlich den Datentyp `Integer`.

2. Nach dem Gregorianischen Kalender ist ein Jahr dann ein Schaltjahr, wenn es ohne Rest durch 4 teilbar ist, es sei denn, es ist auch durch 100 ohne Rest teilbar. In diesem Fall ist es nur dann ein Schaltjahr, wenn es zusätzlich auch durch 400 ohne Rest teilbar ist. Schreiben Sie eine Haskell-Rechenvorschrift `schaltJahr :: Integer -> Bool`, die für eine als Argument übergebene Jahreszahl n.Chr.G. entscheidet, ob es sich bei diesem Jahr um ein Schaltjahr im Gregorianischen Sinn handelt. Der Einfachheit halber nehmen wir dabei an, dass der Gregorianische Kalender bereits ab dem Jahr 1 n.Chr.G. gilt, also ab dem 1. Januar 1.

3. Jahreszahlen n.Chr.G. können in einfacher Weise als ganze Zahlen kodiert werden. So steht die Zahl 20041013 für den 13. Oktober 2004. Entsprechend wird der 1. Februar 1567 durch die Zahl 15670201 dargestellt und der 4. April 444 durch 4440404. Schreiben Sie eine Funktion `ageInDays`, die das Alter eines Menschen an einem bestimm-

ten Tag in Tagen berechnet. Zu diesem Zweck erwartet die Funktion `ageInDays` als erstes Argument den Geburtstag des Menschen, als zweites Argument das Datum, für das das Lebensalter in Tagen berechnet werden soll. Beide Daten sollen einzeln (also nicht ein Paar von Daten) nach obigem Schema kodiert als ganze Zahlen vom Typ `Integer` übergeben werden. Stimmen beide Daten überein, so soll das Ergebnis 1 sein. Liegt das zweite Datum zeitlich vor dem ersten, so soll das Ergebnis 0 sein, ansonsten das entsprechende Lebensalter in Tagen. Berücksichtigen Sie auch Schaltjahre in Ihrer Berechnung und stören Sie sich nicht an außergewöhnlich langen Lebensspannen. Wie die Argumente soll auch das Ergebnis der Funktion `ageInDays` vom Typ `Integer` sein.

4. Im folgenden ist ein Anfangsstück des Pascalschen Dreiecks dargestellt:

```
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5   10  10  5   1
    ...
```

Dieses Dreieck ist wie folgt konstruiert. In der 1. Zeile steht eine 1. Jede weitere Zeile enthält genau eine Zahl mehr als ihre Vorgängerzeile. Ist das Dreieck bis zu einer bestimmten Zeile konstruiert, werden die Zahlen der nachfolgenden Zeile dadurch gebildet, dass sukzessive die Summe zweier benachbarter Zahlen der darüberliegenden Zeile berechnet wird und zusätzlich am Anfang und am Ende der so entstehenden neuen Zeile eine 1 angefügt wird. Schreiben Sie eine Haskell-Rechenvorschrift `pascalTriangle`, die bei Eingabe einer natürlichen Zahl n die n-te Zeile des Pascalschen Dreiecks als Liste ganzer Zahlen ausgibt. Für Werte von n kleiner als 1 soll als Ergebnis die leere Liste abgeliefert werden. Verwenden Sie `Integer` als Typ für das Argument und die Elemente der Resultatliste.