5. Aufgabenblatt zu Funktionale Programmierung vom 13.11.2006. Fällig: 21.11.2006 / 28.11.2006 (jeweils 15:00 Uhr)

Themen: *Typklassen und überladene Funktionen*

Für dieses Aufgabenblatt sollen Sie Haskell-Rechenvorschriften für die Lösung der unten angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens `Aufgabe5.hs` ablegen. Sie sollen für die Lösung dieses Aufgabenblatts also wieder ein "gewöhnliches" Haskell-Skript schreiben. Versehen Sie wieder wie auf den bisherigen Aufgabenblättern alle Funktionen, die Sie zur Lösung brauchen, mit ihren Typdeklarationen und kommentieren Sie Ihre Programme aussagekräftig. Benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten. Denken Sie bei der Kommentierung daran, dass Ihre Lösungen auch Grundlage für das Abgabegespräch am Semesterende sind.

Im einzelnen sollen Sie die im folgenden beschriebenen Problemstellungen bearbeiten.

1. Wir betrachten die folgenden algebraischen Datentypen und Typsynonyme in Haskell:

```
data Quadrat  = Qu Seitenlaenge
data Rechteck = Re Breite Hoehe
data Kreis    = Kr Radius

type Seitenlaenge = Float
type Breite       = Float
type Hoehe        = Float
type Radius       = Float
```

   Schreiben Sie eine Haskell-Typklasse `GeoFigur` mit den Elementen `Quadrat`, `Rechteck` und `Kreis` und den Operationen `umfang` und `flaeche`. Stellen Sie bei der Instanzbildung sicher, dass die überladenen Funktionen `umfang` und `flaeche` angewendet auf Werte vom Typ `Quadrat`, `Rechteck` oder `Kreis` den jeweiligen Umfang bzw. die jeweilige Fläche des Arguments gerundet auf die größte ganze Zahl kleiner oder gleich dem mit Gleitkommazahlen berechneten Wert als Wert des Datentyps `Integer` zurückgeben. Verwenden Sie für die Kreiszahl π in ihrer Berechnung den Näherungswert 3.14.

2. Eine Relation $R \subseteq \mathbb{Z} \times \mathbb{Z}$ über den ganzen Zahlen $\mathbb{Z}$ heißt *antisymmetrisch*, wenn folgendes gilt:

$$\forall\, x, y \in \mathbb{Z}.\; x\, R\, y \;\wedge\; y\, R\, x \;\Rightarrow\; x = y$$

   Für die durch die Adjazenzmatrix eines Graphen (siehe Aufgabenblatt 3) induzierte Relation bedeutet Antisymmetrie, dass es keine Kreise der Länge 2 (gemessen in Anzahl der beteiligten Kanten) gibt. Dazu legen wir hier fest, dass der Eintrag 1 an der Position (m, n) für die Existenz einer gerichteten Kante von m nach n steht und der Eintrag 0 für deren Nichtexistenz.

   Schreiben Sie eine Haskell-Typklasse `Matrix` mit den Elementen `Relation` und `Graph` von Aufgabenblatt 2 und Aufgabenblatt 3 und einer Wahrheitswertfunktion `istAntiSym`, die angewendet auf einen Wert vom Typ `Relation` bzw. `Graph` angibt, ob die Relation antisymmetrisch bzw. der Graph frei von Kreisen der Länge 2 ist. Bei der Implementierung dürfen Sie davon ausgehen, dass die Funktion `istAntiSym` nur mit quadratischen Matrizen über Wahrheitswerten bzw. über den Zahlen 0 und 1 getestet wird.

3. Wir betrachten die folgenden algebraischen Datentypen und Typsynonyme in Haskell:

```
data Kugel   = Ku Durchmesser
data Wuerfel = Wu Kantenlaenge

type Durchmesser  = Float
type Kantenlaenge = Float
```

   Schreiben Sie eine Typklasse `DreiDimFigur` mit dem Kontext `GeoFigur`, den Elementen `Kugel` und `Wuerfel` und der überladenen Funktion `volumen`. Stellen Sie bei der Instanzbildung sicher, dass die überladene Funktion `volumen` angewendet auf Werte vom Typ `Kugel` oder `Wuerfel` das jeweilige

Volumen des Arguments gerundet auf die größte ganze Zahl kleiner oder gleich dem mit Gleitkommazahlen berechneten Wert als Wert des Datentyps `Integer` zurückgibt. Das gilt in gleicher Weise für die von den Funktionen `umfang` und `flaeche` zu liefernden Werte. Der Umfang einer Kugel werde dabei am Äquator gemessen, in analoger Weise werde dies für einen Würfel gemacht. Unter der Fläche einer Kugel und eines Wüfels verstehen wir deren Oberflächen. Verwenden Sie für die Kreiszahl π in ihrer Berechnung wieder den Näherungswert 3.14.

4. Wir definieren auf der Menge der ganzen Zahlen $\mathbb{Z}$ eine eingeschränkte Additionsoperation $\oplus : \mathbb{Z} \times \mathbb{Z} \to I\!N_0$ wie folgt:

$$\forall\, m, n \in \mathbb{Z}.\ m \oplus n =_{df} \begin{cases} \text{m + n} & \text{falls } m, n \geq 0 \\ 0 & \text{sonst} \end{cases}$$

   Dabei bezeichnet das Symbol + die übliche Addition auf natürlichen Zahlen (einschließlich der 0) und das Symbol $=_{df}$ steht für "definitionsgemäß gleich".

   Schreiben Sie eine Haskell-Rechenvorschrift `addMin3` mit der Signatur `addMin3 :: String -> String -> String`, die die Additionsoperation $\oplus$ für (-3)-adische Zahlen realisiert. Die Rechnung soll dabei ausschließlich im (-3)-adischen System ausgeführt werden, nicht also etwa durch Umwandlung der Argumente in ein anderes b-adisches Zahlsystem, Ausführung der Addition in diesem System und Rückumwandlung des Resultats in die entsprechende (-3)-adische Repräsentation.

Bei der Implementierung können Sie davon ausgehen, dass alle Funktionen nur mit "passenden" Argumenten aufgerufen werden, die Funktion `addMin3` etwa nur mit nichtleeren Zeichenreihen über den Zeichen '0', '1' und '2'.