

Aufgabe 3:**Klassen und Instanzen****5 Punkte**

Der folgende Datentyp `Konto` dient zur Beschreibung eines Kontostandes und wird wie folgt definiert.

```
data Konto = Anlegen | Ausgaben Int Int Konto | Einnahmen Int Int Konto
```

Dabei wird durch den Konstruktor `Anlegen` ein `Konto` (mit dem Kontostand 0) angelegt. Die Konstruktoren `Ausgaben` und `Einnahmen` haben jeweils zwei Parameter für Euro und Cent und beziehen sich auf ein vorhandenes `Konto`. Beachten Sie, dass bei `Ausgaben` der durch die Parameter übergebene Wert vom aktuellen Kontostand abzuziehen ist.

Implementieren Sie `Konto` als Instanz der Klassen `Eq` und `Show`, wobei sich die Gleichheit auf den wirklichen Kontostand und nicht auf die Art seiner Entstehung beziehen soll. Bei `Show` sollten Strings der Form

‘‘Kontostand: 3,75 Euro’’ oder ‘‘Kontostand: -15,05 Euro’’ erzeugt werden.

Hinweis: Kontostände intern als reine Centbeträge verwalten.

Aufgabe 4:**Fehlerkorrigierende Codes****1 + 3 + 2 Punkte**

Die folgende Codierung $c : \mathbb{B}^4 \longrightarrow \mathbb{B}^7$ ($\mathbb{B} = \{0, 1\}$) nutzt drei spezielle Paritätsbits :

$$c(b_1, b_2, b_3, b_4) := (b_1, b_2, b_3, b_4, p_1, p_2, p_3) \quad \text{wobei} \quad \begin{aligned} p_1 &:= (b_2 + b_3 + b_4) \pmod 2 \\ p_2 &:= (b_1 + b_3 + b_4) \pmod 2 \\ p_3 &:= (b_1 + b_2 + b_4) \pmod 2 \end{aligned}$$

- Sei der Code C das Bild der Funktion c . Zeigen Sie, dass der Minimalabstand von C kleiner oder gleich 3 ist (was muss man dazu zeigen?)!
- Der Minimalabstand von C ist gleich 3. Man beweist das mit einer Fallunterscheidung bezüglich des Abstands der zu codierenden 4-Tupel. Analysieren Sie (nur!) für den Fall von zwei 4-Tupeln mit Abstand 1, welchen Hammingabstand die abgeleiteten Codewörter haben können.
- Die 7-Tupel $w_1 = (1, 0, 0, 1, 1, 1, 0)$ und $w_2 = (0, 1, 1, 0, 1, 1, 0)$ haben jeweils den Abstand 1 zu einem Codewort. Bestimmen Sie diese nächsten Codewörter (Fehlerkorrektur).

Aufgabe 5:**Präfixcodes und binäre Bäume****3 + 3 Punkte**

Der algebraische Datentyp `codeTree` wird wie folgt definiert:

```
data codeTree = leaf Char | Node codeTree codeTree
```

Offensichtlich sind nur die Blätter mit codierten Zeichen (`Char`) bewertet und wie üblich werden Wege von der Wurzel zu einem Blatt mit $\{0, 1\}$ -Strings bewertet (linker Teilbaum 0 / rechter Teilbaum 1).

- Definieren Sie zwei Funktionen `search`, `unique :: Char -> codeTree -> Bool`, die feststellen, ob ein gegebenes Zeichen in einem Blatt bzw. in genau einem Blatt des Baums auftaucht.
- Definieren Sie eine Funktion `code :: Char -> codeTree -> String`, die für ein gegebenes Zeichen, das im Baum repräsentierte Codewort ausgibt. Sie können davon ausgehen, dass die Funktion nur für eindeutig codierte Zeichen aufgerufen wird (d.h. Fehlermeldung bei Zeichen, die nicht oder mehrfach im Baum auftreten nicht notwendig).