Prof. Dr. Manfred Schmidt-Schauß
Institut für Informatik

Fachbereich Informatik und Mathematik/ Institut für Informatik
Johann Wolfgang Goethe-Universität Frankfurt am Main

Einführung in die funktionale Programmierung

Wintersemester 2025/2026

Aufgabenblatt Nr. 5

Abgabe: Dienstag 13. Januar 2026 bis 10:00, Besprechung 16. Januar 2026

Das Aufgabenblatt hat 50 Punkte (und 25 mögliche Bonuspunkte)

Aufgabe 1 (20 Punkte)

Der Datentyp `Expr a` zur Darstellung von KFPT-Ausdrücken ist:

```
> data Expr a =
>    Var a                                    -- x       = Var "x"
>  | App (Expr a) (Expr a)                    -- (e1 e2) = App e1 e2
>  | Lam a (Expr a)                           -- \x.e    = Lam "x" e
>  | ListCons (Expr a) (Expr a)               -- a:as    = ListCons a as
>  | ListNil                                  -- []      = ListNil
>  | BoolTrue                                 -- True    = BoolTrue
>  | BoolFalse                                -- False   = BoolFalse
>  | CaseList (Expr a) (Expr a) (a,a,Expr a)-- case_List e of {[] -> e1; (x:xs) -> e2)
>                                             -- = CaseList e e1 ("x",xs",e2)
>  | CaseBool (Expr a) (Expr a) (Expr a)    -- case_Bool e of {True -> e1; False -> e2)
>                                             -- = CaseBool e e1 e2
```

Sei der Typ für Variablen definiert als

```
> newtype Varname = Varname String
```

Definieren sie je eine Instanz der Typklasse `Show` für den Typ `Expr a` und den Typ `Varname`, sodass der Ausgabestring der `show`-Funktion für jeden Ausdruck vom Typ `Expr Varname` einen syntaktisch korrekten Haskell-Ausdruck gleicher Bedeutung darstellt (z.B. werden Abstraktionen als `\x -> e` dargestellt).

Da der Typ `Expr` polymorph über Variablennamen ist, müssen Sie bei der Instanzdefinition zusatzlich als Klassenbedingung fordern, dass bereits eine `Show`-Instanz für `a` existiert, d.h. die Instanzdefinition beginnt mit

```
instance Show a => Show (Expr a) where
```

Vorsicht: `ListCons (Expr a) (Expr a)` statt `ListCons (Expr a) [(Expr a)]` benutzen...

Aufgabe 2 (30 Punkte)

Polymorphe binäre Bäume `BBaum` und n-äre Bäume `NBaum` seien in Haskell definiert als:

```
> data BBaum a = BBlatt a                       -- Blatt mit Markierung
>              | BKnoten a (BBaum a) (BBaum a) -- Markierung, linker u. rechter Teilbaum
>  deriving(Show)

> data NBaum a = NBlatt a              -- Blatt mit Markierung
>              | NKnoten a [NBaum a]   -- Markierung und Liste der Kinder
>  deriving(Show)
```

a) Definieren Sie in Haskell eine Konstruktorklasse `IstBaum`, die Operationen für Baum-artige Datentypen überlädt. Als Klassenmethoden sollen dabei zur Verfügung stehen:

   - `teilbaeume` liefert die Liste der Unterbäume der Wurzel.
   - `istBlatt` testet, ob ein Baum nur aus einem Blatt besteht und liefert dementsprechend `True` oder `False`. (8 Punkte)

b) Definieren Sie eine Unterklasse `IstMarkierterBaum` von `IstBaum`, die Operatoren für Bäume mit Knotenmarkierungen überlädt. Die Klassenmethoden sind:

   - `markierung` liefert die Beschriftung der Wurzel.
   - `knoten` liefert alle Knotenmarkierungen eines Baums als Liste.
   - `kanten` liefert alle Kanten des Baumes, wobei eine Kante als Paar von Knotenmarkierungen dargestellt wird.

   Geben Sie dabei Default-Implementierungen für `knoten` und `kanten` innerhalb der Klassendefinition an. (12 Punkte)

c) Implementieren Sie Instanzen der Klassen `IstBaum` und `IstMarkierterBaum` jeweils für die Datentypen `BBaum` und `NBaum`. (10 Punkte)

Für die beiden folgenden Beispielbäume:

```
> bspBBaum = BKnoten 1 (BKnoten 2 (BBlatt 3) (BBlatt 4)) (BKnoten 5 (BBlatt 6) (BBlatt 7))
> bspNBaum = NKnoten 1 [NKnoten 2 [NBlatt 3,NBlatt 4, NBlatt 5]
>                      ,NKnoten 6 [NBlatt 7, NKnoten 8 [NBlatt 9, NBlatt 10]]]
```

verdeutlichen die folgenden Beispielaufrufe die geforderten Funktionalitäten:

```
*Main> teilbaeume bspBBaum
[BKnoten 2 (BBlatt 3) (BBlatt 4),BKnoten 5 (BBlatt 6) (BBlatt 7)]
*Main> teilbaeume bspNBaum
[NKnoten 2 [NBlatt 3,NBlatt 4,NBlatt 5],NKnoten 6 [NBlatt 7,NKnoten 8 [NBlatt 9,NBlatt 10]]]
*Main> knoten bspBBaum
[1,2,3,4,5,6,7]
*Main> knoten bspNBaum
[1,2,3,4,5,6,7,8,9,10]
*Main> kanten bspBBaum
[(1,2),(1,5),(2,3),(2,4),(5,6),(5,7)]
*Main> kanten bspNBaum
[(1,2),(1,6),(2,3),(2,4),(2,5),(6,7),(6,8),(8,9),(8,10)]
*Main> map markierung (teilbaeume bspNBaum)
[2,6]
*Main> map markierung (teilbaeume bspBBaum)
[2,5]
```

Aufgabe 3 (25 Punkte) Magische Quadrate (Zusatz-Aufgabe)

In der Literatur (und in der Vorlesung) gibt es Entdeckungen und Untersuchung zu Zahlenquadraten (sog. magische Quadrate) von natürlichen Zahlen mit einigen extra-Eigenschaften. (Schon von Albrecht Dürer überliefert).

Schreiben Sie ein Programm, dass zu einem gegebenen (geraden,) $n \geq 4$ magische Quadrate (als Matrizen, d.h. Listen von Listen von positiven Integers) erzeugen kann.

1. Erzeugung initiales Quadrat zu n mit den Zahlen $1, \ldots, n^2$.
2. (Mehrere) Permutationsoperationen auf dem Quadrat:
   (a) Vertauschen von Zeilen; z.B. $1 \leftrightarrow 2, \ldots, n-1 \leftrightarrow n$ und $1 \leftrightarrow n, 2 \leftrightarrow n-1, \ldots$.
   (b) Vertauschen von Spalten; (analog)
   (c) **Ergänzung**: Invertieren einer Zeile / einer Spalte: in der Zeile bzw. Spalte Reihenfolge umkehren.

Schreiben Sie auch einige Funktionen, die die Eigenschaften der erzeugten Quadrate berechnet bzw. testet: (Mindestens 2 Tests im Fall $n = 8$) ob die erzeugten magischen Quadrate wirklich "magisch" sind (Z.B. Summe der Zahlen in Zeilen bzw. Spalten und bestimmten Teilquadraten). und dokumentieren Sie die Ergebnisse.

- Summe der Zahlen in Zeilen / Spalten bzw. (sollte durch die Konstruktion immer stimmen).
- Summe der Zahlen in Teilquadraten $n/2 \times n/2$ oder von Teilquadraten $n/4 \times n/4$. (hier muss n durch 4 teilbar sein.
- Summe der Zahlen in anderen Substrukturen? Welche sind sinnvoll?

Anmerkungen und Hinweise nachträglich zu Aufgabe 3: (am 15.12.2025)

- Spalten- und Zeilen-Invertierung als Hinweis ergänzt
- Empfehlung: Algorithmische Baumsuche: Am besten eine Veränderungen machen und dann Eigenschaften testen usw.

Zum Beispiel kann man das klassische 4×4 magische Quadrat (ein äquivalentes) berechnen, indem man im Start-Quadrat mit den Zahlen 1,. . . ,16:

1. die beiden mittleren Spalten invertiert
2. die beiden mittleren Zeilen tauscht;
3. die beiden mittleren Zeilen invertiert.