

Funktionale Programmierung

6. Übungsblatt

Prof. Dr. Margarita Esponda

Thema: Algebraische Datentypen**1. Aufgabe** (3 Punkte)

Seien folgende algebraische Datentypen und Funktionen, die in der Vorlesung definiert worden sind:

```
data B = T | F deriving Show
data Nat = Zero | S Nat deriving Show
data ZInt = Z Nat Nat deriving Show
```

a) (15 P.) Vervollständigen Sie, die in der Vorlesung besprochenen Funktionen für die algebraischen Datentypen **B** und **Nat** um folgenden Funktionen:

```
eqB :: B -> B -> B      -- testet nach Gleichheit          (1 P.)
xorB :: B -> B -> B      -- exclusives Oder                (1 P.)
(=>>) :: B -> B -> B.    -- Implikation                    (2 P.)

eqN :: Nat -> Nat -> B    -- mit einer rekursiven Definition  (1 P.)
evenN :: Nat -> B        -- überprüft, ob die Zahl gerade ist (2 P.)
isDivisorN :: Nat -> Nat -> B -- überprüft, ob die zweite Zahl die erste Zahl genau teilt (3 P.)

halbN :: Nat -> Nat      -- ganzzahlige Division durch Zwei  (2 P.)
ggtN :: Nat -> Nat -> Nat -- größter gemeinsamer Teiler    (3 P.)
```

b) (4 P.) Definieren Sie für die algebraische Datentypen **B** und **Nat** folgende Haskell-Funktion, die die maximale Anzahl von Pizza-Teilen berechnet, wenn die Pizza mit **n** geraden Schnitten aufgeteilt wird?

```
maxSurfaces :: Int -> Int
maxSurfaces 0 = 1
maxSurfaces (n+1) = (maxSurfaces n) + n + 1
```

c) (11 P.) Programmieren Sie folgende Funktionen für den algebraischen Datentyp **ZInt** aus der Vorlesung. Sie dürfen, bei Bedarf, nur die **error**-Funktion von Haskell verwenden.

```
neqZ :: ZInt -> ZInt -> B      -- testet nach Gleichheit
(>>>) :: ZInt -> ZInt -> B      -- testet ob die 1. Zahl kleiner als die 2. Zahl ist.
negZ :: ZInt -> ZInt           -- berechnet die negative Zahl
multZ :: ZInt -> ZInt -> ZInt   -- multipliziert zwei Zahlen
absZ :: ZInt -> ZInt           -- absoluter Wert einer Zahl
powZ :: ZInt -> Nat -> ZInt     -- Potenz
isDivisorZ :: ZInt -> ZInt -> B -- überprüft ob die 2. Zahl Teiler der 1. Zahl ist
```

Wichtige Hinweise:

- 1) Verwenden Sie geeignete Namen für Ihre Variablen und Funktionsnamen, die den semantischen Inhalt der Variablen oder die Semantik der Funktionen wiedergeben.
- 2) Verwenden Sie vorgegebene Funktionsnamen, falls diese angegeben werden.
- 3) Kommentieren Sie Ihre Programme.
- 4) Verwenden Sie geeignete lokale Funktionen und Hilfsfunktionen in Ihren Funktionsdefinitionen.
- 5) Geben Sie für alle Funktionen die entsprechende Signatur an.
- 6) Schreiben Sie getrennte Test-Funktionen für alle Aufgaben.
- 7) Die Lösungen sollen elektronisch (nur Whiteboard-Upload) abgegeben werden. **Keine verspätete Abgabe per Email ist erlaubt.**