Name: _____

0.1 Funktionale Programmierung

Bäume gehören zu den wichtigsten in der Informatik auftretenden Datenstrukturen. In dieser Aufgabe verwenden wir den Datentyp Tree zur Darstellung von Binärbäumen, die an den Blättern mit Werten markiert sind und an den Knoten mit Operationen:

```
data Tree =
        Value Integer
      | Mark Op Tree Tree
      deriving (Eq, Show)

data Op = Plus | Mult | Minus
      derivin (Eq, Show)
```

Schreiben Sie eine Haskell-Funktion *balanced*, die einen Baum als Eingabe entgegennimmt und entscheidet, ob in diesem Baum genausoviele positive Werte als Blätter vorkommen wie negative. Die Null soll dabei weder als positiv noch negativ zählen. *(6 Punkte)*

Beispiel:

```
balanced (Mark Plus (Value 3) (Mark Minus (Value 0) (Value (-4)))) == True
balanced (Mark Plus (Value 3) (Mark Minus (Value 2) (Value (-4)))) == False

balanced :: Tree -> Bool
```

0.2 Funktionale Programmierung

Schreiben Sie die Funktionen *laenge*, die die Länge einer Liste berechnet, und *summe*, die die Summe der ersten n Zahlen berechnet. Beachten Sie, dass alle Eingaben behandelt werden müssen und geben Sie die Signatur der Funktionen an. *(4 Punkte)*
Beispiel:

```
laenge [[], [1,2], [2]]     -- => 3
summe 4                     -- => 10
```

0.3 Funktionale Programmierung

Gegeben sei folgender Datentyp *Zeit*, der die Menge von Zeitdauer-Angaben im Format *Wochen : Tage : Stunden* modelliert.

```
data Zeit = WTS Integer Integer Integer
```

Hierbei gilt, dass die Angaben normiert vorliegen, das heißt alle Komponenten Werte größer gleich 0 beinhalten und die Komponenten Tage und Stunden nur Werte kleiner 7 bzw. 24 enthalten.

(a) Schreiben Sie eine Funktion *inStunden*, die eine Zeit als Parameter entgegennimmt und diese in die Anzahl von Stunden umwandelt. *(2 Punkte)*

```
inStunden :: Zeit -> Integer
```

(b) Schreiben Sie eine Funktion *alsZeit*, die eine ganze Zahl, die eine Anzahl von Stunden darstellt, als Parameter entgegennimmt und diese in eine Zeit umwandelt. *(2 Punkte)*

```
alsZeit :: Integer -> Zeit
```