(FP-Aufgaben 11)

Aufgabe 1: Überschreiben Sie die (==)-Funktion für Binärbäume. Wobei ein Binärbaum durch den folgenden Typ gegeben ist:

```
data Baum a = Knoten a (Baum a) (Baum a)
            | Leer
```

Aufgabe 2: Implementieren Sie einen kleinen Taschenrechner für den Typ „Ausdruck“.

Sie können den folgenden Datendeklaration erweitern und als einen Datentyp verwenden:

```
data Ausdruck a = Zahl a
                | Add (Ausdruck a) (Ausdruck a)      -- Addition von zwei Ausdrücken
                | Sub (Ausdruck a) (Ausdruck a)      -- Subtraktion von zwei Ausdrücken
```

Als Beispiel:

Der Aufruf

```
rechne (Add (Zahl 1.5) (Add (Zahl 3) (Zahl 2)))
```

sollte (1.5 + (3 + 2)) auswerten und 6.5 als ein Ergebnis zurückliefern.

Hinweis:

a) Definieren Sie den Typ „Ausdruck“ als eine Instanz der folgenden Klasse „Rechner“ und implementieren Sie die Funktion „*rechne*“.

```
class Rechner a where
    rechne :: a -> Double
```

b) Schreiben Sie Aufruf-Beispiele und testen Sie Ihr Programm.