Tupel

In Pascal werden mehrere Komponenten in RECORDS verpackt. Java kapselt zusammengehörende Daten in Klassen. In der Mathematik fasst man mehrere Komponenten in Tupel oder Vektoren zusammen. Funktionale Sprachen erlauben beliebig große Tupel. Synonyme machen die Sprache verständlicher. Durch Musteranpassung können die Komponenten einfach bearbeitet werden.

Beispiel:

```
type Person = (String, String, Int)   -- Name, Vorname, Alter

p1 :: Person
p1 = („Bauer“, „Tom“, 18)

name :: Person -> String              -- Selektion Name
name (n,v,a) = n

vorname :: Person -> String           -- Selektion Vorname
vorname (n,v,a) = v

alter :: Person -> Int                -- Selektion Alter
alter (n,v,a) = a
```

Übungen

1. Eine Funktion `abstand0` berechnet den Abstand eines Punktes P vom Nullpunkt, die Funktion `abstand` den Abstand zweier Punkte.

   Hinweis: Punkte sind als Tupel aus `Float`-Koordinaten zu deklarieren. Die Wurzelfunktion heißt `sqrt`.

2. Gesucht sind Funktionen, die zwei Personen vergleichen.
   a) `gleicherName` liefert `True`, wenn die Namen gleich sind.
   b) `gleichesAlter` liefert `True`, wenn die Personen gleich alt sind.

3. Die letzte Aufgabe kann eleganter gelöst werden, wenn als zusätzlicher Parameter eine passende Projektionsfunktion mitgenutzt wird:

   ```
   gleich name(„Bauer“, „Tom“, 18) („Bauer“, „Dora“, 17)
   ```

   Der erste Parameter `name` ist die oben definierte Funktion, die einen `String` zurückliefert. Ebenso kann `alter` mit dem Rückgabetyp `Int` benutzt werden. Solche universelle Funktionen heißen polymorph.

   Schreiben Sie die Funktion `gleich`, die prüft, ob die übergebene Funktion f für beide Personen zum selben Ergebnis führt.

   Tipp: Die Typendefinition sieht folgendermaßen aus:
   ```
   gleich :: Eq a => (Person -> a) -> Person -> Person -> Bool
   ```

   Überlegen Sie, wie diese Typendefinition zustande kommt und was sie bedeutet.