Die bei imperativen Programmiersprachen üblichen Verzweigungen innerhalb eines Programms zur Fallunterscheidung sind auch in Haskell möglich. Dabei gibt es mehrere Varianten, die je nach Aufgabe mehr oder weniger sinnvoll eingesetzt werden können.

Diskrete Lösung

Die Funktion wird in der diskreten Lösung für alle möglichen Eingabewerte definiert.

Beispiel:

```
and :: Bool -> Bool -> Bool
and False False = False
and True False = False
and False True = False
and True True = True
```

Wächterlösung

Eine übersichtlichere Form der Fallunterscheidungen bieten sogenannte Wächter | . Diese werden in der Auswertung von oben nach unten abgearbeitet. Dabei wird der erste passende Fall benutzt. Für einen Restfall kann auch das Schlüsselwort `otherwise` genutzt werden. Die folgende Funktion bestimmt das Vorzeichen (sign) einer Zahl:

Beispiel:

```
sgn :: Float -> Int
sgn x
      | x<0           = -1
      | x==0          = 0
      | otherwise     = 1
```

Musteranpassung (pattern matching)

Häufig schreibt man einige Sonderfälle auf und formuliert dann einen allgemeinen Fall. Diese Technik nennt man im Englischen „pattern matching“ (Musteranpassung). Beim Aufruf prüft der Interpreter, ob ein Sonderfall vorliegt. Trifft dies nicht zu, wird der allgemeine Fall genutzt.

Beispiel:

```
antwort „“ = „leerer String“
antwort "
```

Übungen

1. Definieren Sie eine Funktion `istVokal`, die überprüft, ob ein Buchstabe ein Vokal ist. Benutzen Sie hierzu die Technik der Musteranpassung. Beachten Sie die Groß- und Kleinschreibung.

2. Eine Funktion `begruessung` ermittelt anhand der übergebenen Stundenangabe eine passende Begrüßung.

   Beispiel:

   ```
   begruessung 9     --> „Guten Morgen!“
   begruessung 21    --> „Guten Abend!“
   ```