Hoogle

- `http://www.haskell.org/hoogle/`
- Suchmaschine für Haskell-Funktionen
- Suche sowohl nach Funktionsnamen als auch **Funktionssignatur**
- Bsp.: Suchen eine Funktion, die einen beliebiges Argument n-mal repliziert und in eine Liste einfügt
- Wie könnte die Signatur aussehen?

Hoogle

- Suchen nach `Integer -> a -> [a]`
- Erstes Resultat: `intersperse` – falsches Resultat
- Zweites Resultat: `genericReplicate` – richtiges Resultat!
  Tatsächliche Signatur: `Integral i => i -> a -> [a]`
- Braucht `import Data.List`
- Weiter unten: Variante für `Int`: `replicate`
- Die meisten Funktionen operieren auf `Int`, nicht `Integer`.
  Für die Übung wird meistens `Integer` verlangt – daher:
- Tipp: Für viele Funktionen gibt es eine "allgemeine" `Integral`-Variante mit dem Präfix "generic" – z.B. `genericLength`

Backticks

- Backticks (= accent grave): Für Funktionen mit zwei Argumenten
- Wandeln die Funktion in Infix-Variante um
- Statt: `mod 15 4 (Ergebnis: 3)`
- folgendes: `15 ‘mod‘ 4 (ebenfalls 3)`
- (Achtung “Backticks”: Sehen hier in LaTeX leider wie normale Hochkommas aus – funktioniert aber nur mit echten Backticks)

Klammerung

- Schont eure Nerven und klammert jeden Ausdruck
- Tw. kommen sonst schwer durchschaubare Fehler:

```
foo :: Integer -> Integer
foo n | (n > 3) = n+3 : n+6 : []

... Couldn't match expected type ‘Integer’
with actual type ‘[a0]’
```

Klammerung

. . . oder nicht einmal Fehlermeldungen – einfach falsches Resultat:

```
binom n k = (fac n) `div` ((fac k) * (fac n-k))
                                        ^^^
```

```
5 `binom` 0 ->> 1
5 `binom` 1 ->> 1
5 `binom` 2 ->> 0
...
```

Debug.Trace

- Idee: “Logging” als Debugging-Hilfe
- Zum Verwenden: `import Debug.Trace`

Beispiel:

```
import Debug.Trace

fac :: Integer -> Integer
fac 0 = 1
fac n = trace ("bis jetzt: " ++ (show res)) res
  where
    res = n * (fac (n-1))
```

Debug.Trace

WARNUNGEN:

- Debug.Trace ist *nicht* als Ausgabemodul gedacht!
- Aufrufreihenfolge ist i.A. nicht vorhersehbar – nur verwenden, wenn man *verstehen* will, wie Haskell eine Funktion abarbeitet
- Debug.Trace führt relativ schnell zu undurchschaubarem Code
- Folgender Ansatz ist *praktisch immer* schneller als Loggen mit Debug.Trace:
  - Sauberer Programmierstil (Klammerung!)
  - Funktionen in kleinen Schritten schreiben
  - Jeden Schritt einzeln testen
  - Nach und nach zusammenfügen
- Debug.Trace kann in Extremfällen hilfreich sein.
- Wer oft versucht ist, das Modul zu verwenden, sollte seine Programmierstrategie überdenken