# Übungen zu Softwareentwicklung III, Funktionale Programmierung

Blatt 7, Woche 8

Leonie Dreschler-Fischer

WS 2017/2018

**Ausgabe:** Freitag, 8.12.2017,

**Abgabe der Lösungen:** bis Montag, 18.12.2017, 12:00 Uhr per email bei den Übungsgruppenleitern.

**Ziel: Rekursion:** Die Aufgaben auf diesem Zettel dienen dazu, sich mit dem Entwurf von endrekursiven Funktionen und dem Einsatz von Funktionen höherer Ordnung vertraut zu machen.

**Bearbeitungsdauer:** Die Bearbeitung sollte insgesamt nicht länger als 5 Stunden dauern.

**Homepage:**
http://kogs-www.informatik.uni-hamburg.de/~dreschle/teaching/Uebungen_Se_III/Uebungen_Se_III.html

Bitte denken Sie daran, auf den von Ihnen eingereichten Lösungsvorschlägen *Ihren Namen und die Matrikelnummer, den Namen der Übungsgruppenleiterin / des Übungsgruppenleiters* und *Wochentag und Uhrzeit der Übungsgruppe* anzugeben, damit wir ihre Ausarbeitungen eindeutig zuordnen können.

# 1 Abbilden

Bearbeitungszeit 1 Std., 10 Pnkt.

Definieren Sie eine Funktion `zaehlen`, die einen Wert `x` sowie eine Liste von Werten `xs` als Argumente annimmt und ermittelt, wie oft der Wert in der Liste vorkommt. Hier ein Beispiel:

```
> (zaehlen 3 '(2 4 3 2 3 4 1)) ⟶ 2
```

Programmieren Sie diese Funktion in drei Varianten:

- als allgemein rekursive Funktion,
- als endrekursive Funktion und
- mittels geeigneter Funktionen höherer Ordnung.

# 2 Das Spiel des Lebens

Bearbeitungszeit 4 Std., ingesamt 30 Punkte

In dieser Aufgabe sollen Sie das sogenannte "Spiel des Lebens" in Racket implementieren. Das Spiel (im Original: Game Of Life) wurde von dem Mathematiker John Horton Conway 1970 erdacht und basiert auf einem zweidimensionalen zellulärem Automaten. Das Spielfeld besteht aus einem $N \times N$ großen Raster, wir legen für diese Aufgabe $N = 30$ fest.

Zu Beginn des Spiels wird ein Startzustand festgelegt. Von diesem ausgehend ergibt sich der Folgezustand anhand einfacher Regeln aus dem vorherigen Zustand. Hierbei ist zu berücksichtigen, dass jede Zelle (außer die Zellen am Rand[1]) von 8 weiteren umgeben ist. Diese werden im Folgenden "Nachbarn der Zelle" genannt. Eine Zelle kann entweder als "tot" oder "lebendig" markiert sein. Die Regeln sind:

- Eine tote Zelle mit genau drei lebenden Nachbarn wird (neu) geboren,
- eine lebende Zelle mit weniger als zwei lebendigen Nachbarn stirbt an Einsamkeit, und
- eine lebende Zelle mit mehr als drei lebendigen Nachbarn stirbt wegen Überbevölkerung.

[1] Alle Zellen außerhalb des Spielfelds werden vorerst als "tot" markiert angenommen.

Eine lebende Zelle mit zwei oder drei lebenden Nachbarn bleibt in der Folgegeneration übrigens am Leben, Ihr Zustand bleibt somit unverändert. Ebenso verhält es sich für alle anderen, hier nicht weiter aufgezählten Konfigurationen von Zellen und Ihren Nachbarschaften.

*Verwenden Sie für die folgenden Aufgaben Funktionen höherer Ordnung, wann immer es Ihnen hilfreich erscheint!*

## 2.1 Modellierung des Spielzustands

5 Pnkt.

Entwerfen Sie eine Datenstruktur für den Spielzustand des Spiel des Lebens. Beachten Sie hierbei die verschiedenen Operationen, die während des Spielablaufs an dieser Datenstruktur durchgeführt werden müssen und *begründen Sie Ihren Entwurf*!

## 2.2 Visualisierung des Spielzustands

5 Pnkt.

Schreiben Sie, ausgehend von Ihrer Datenstruktur aus Aufgabe 2.1, eine Funktion, die Ihren Spielzustand mithilfe des Pakets `(require 2htdp/image)` in ein Bild überführt. Für ein $30 \times 30$ großes Spielfeld sollten $10 \times 10$ Pixel große Quadrate pro Eintrag eine angemessene Größe darstellen. Verwenden Sie als Kodierung für tote Zellen durchsichtige, schwarz umrandete Quadrate und für lebendige Zellen schwarz ausgefüllte Quadrate.

## 2.3 Spiellogik und Tests

15 Pnkt.

Um die Spiellogik abbilden zu können, benötigen Sie einige Funktionen, die aufgrund des aktuellen Spielzustands den folgenden Spielzustand ermitteln. Schreiben Sie daher Funktionen, die folgendes leisten:

- eine Funktion, die für einen beliebigen Index des Spielzustands, z.B. $(x = 20, y = 10)$, die Werte der 8er-Nachbarschaft ermittelt,
- eine Funktion, die die Werte dieser 8er-Nachbarschaft erhält, und gemäß den Spielregeln den Folgezustand des Automaten am übergebenen Index bestimmt, sowie
- eine Funktion, die einen kompletten Spielzustand gemäß der Regeln in einen neuen Spielzustand überführt.

Testen Sie Ihre Funktionen mit einigen bekannten Mustern, die Zyklen im Spiel des Lebens erzeugen, oder anderen besonderen Mustern, wie Gleitern oder Raumschiffen. Eine Vielzahl an Beispielen finden Sie unter:
https://de.wikipedia.org/wiki/Conways_Spiel_des_Lebens

## 2.4 Simulation

5 Pnkt.

Machen Sie sich mit dem World/Universe-Framework vertraut. Sie finden weitere Informationen hierzu in der Racket-Hilfe. Binden Sie es schließlich mit `(require 2htdp/universe)` ein und simulieren Sie das Spiel des Lebens mit 4 Zustandsübergängen pro Sekunde.

Verwenden Sie hierzu Ihren bisher definierten Spielzustand als Startzustand. Binden sie weiterhin die bisher definierten Funktionen zum Zeichnen und Übergang eines Spielzustandes an die Methoden `to-draw` beziehungsweise `on-tick`. Im Anschluss erwecken Sie die Welt mit der Funktion `big-bang` zum Leben.

## 2.5 Zusatzaufgabe: Spielfeldrand

5 Zusatzpnkt.

Schreiben Sie geeignete Teile Ihrer Implementierung so neu, dass statt der Annahme "tot" markierter Zellen am Rand, eine Torus-artige Annahme vorliegt. Am linken Rand beginnt das Spielfeld von rechts kommend, am oberen Rand von unten und vice versa.

Können Sie Unterschiede im Spielablauf für die in Aufgabe 2.3 verwendeten Tests ausmachen? Wenn ja, welche?

**Erreichbare Punkte:** 40

**Erreichbare Zusatzunkte:** 5