Übungen zu Softwareentwicklung III, Funktionale Programmierung

Blatt 7, Woche 8

Leonie Dreschler-Fischer

WS 2016/2017

Ausgabe: Freitag, 9.12.2016,

Abgabe der Lösungen: bis Montag, 19.12.2016, 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 einfachen 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., 9 Pnkt.

Definieren Sie eine Funktion *produkt*, die eine Liste von Zahlen n sowie einen Faktor f als Argument nimmt und eine neue Liste errechnet, die anstelle von n jeweils das Produkt $n \cdot f$ enthält. Beispiel:

> (produkt '(2 4 3) 3) ⟶ '(6 12 9)

Programmieren Sie diese Funktion in drei Varianten:

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

2 Sieben-Segment Anzeige

Bearbeitungszeit 4 Std., ingesamt 26 Punkte

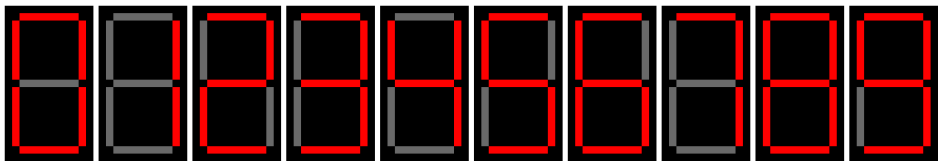
In dieser Aufgabe sollen Sie die Darstellung von Zahlen mittels einer Sieben-Segment Anzeige modellieren. Diese Art von digitalen Displays besteht aus aus insgesamt 7 Leuchtdioden. Je nachdem welche der Dioden aktiv sind, werden (vereinfacht) Zahlen für einen Betrachter lesbar dargestellt.

Verwenden Sie nach Möglichkeit Funktionen höherer Ordnung!

2.1 Zustände

7 Pnkt.

Überlegen Sie sich eine Kodierung des Zustands einer Sieben-Segment Anzeige unter der Annahme, dass beliebig viele Leuchtdioden gleichzeitig leuchten können. Erstellen Sie anschließend eine Liste, die alle möglichen Zustände für die Zahlen 0-9 in sequenzieller Abfolge enthält. Orientieren Sie sich hierbei an folgender Grafik:



2.2 Grafische Darstellung

6 Pnkt.

Schreiben Sie unter Zuhilfenahme des image-Paketes *(require 2htdp/image)* eine Funktion, die ein Sieben-Segment grafisch darstellt. Benutzen Sie ein Rechteck der Größe 100x200 Pixel für den Rahmen und Segmente der Größe 10x80 bzw. 80x10 Pixel für die einzelnen Segmente. Als Farbe wählen Sie bitte *"Red"* für aktive Dioden und *"DimGray"* für inaktive Dioden.

2.3 Simulation 1: Einzelanzeige

3 Pnkt.

Schreiben Sie eine Funktion *(zeige-7segment t)*, die eine Sieben-Segment Anzeige zu einem Zeitpunkt t anzeigt. Mit jedem 28. t (also bei $t = 28, 56, 84, ...$) sollte die Sieben-Segment Anzeige ihren Zustand zum nächsthöheren wechseln. Falls der höchste Zustand überschritten ist, soll wieder von vorne begonnen werden.

Mit dem Animate-Framework *(require 2htdp/universe)* sollten Sie nun ein einzelnes Segment (mit 28 Ticks pro Sekunde) simulieren können:

```
(animate zeige-7segment)
```

2.4 Simulation 2: Zeitmessungsanzeige

10 Pnkt.

Üblicherweise tauchen Sieben-Segment Anzeigen nicht alleine auf. Stellen Sie daher die bereits vergangenen Zeit der Simulation durch die Verknüpfung von sechs Sieben-Segment Anzeigen dar. Hierbei stellen die ersten zwei Anzeigen die Stunden, die mittleren zwei Anzeigen die Minuten und die hinteren zwei Anzeigen die vergangenen Sekunden dar. Ihre Modellierung sollte auch darauf Rücksicht nehmen, dass Programme möglicherweise länger als einen Tag laufen. Entwerfen und skizzieren Sie die Übergänge zwischen den einzelnen Sieben-Segment Anzeigen und implementieren Sie dann eine Funktion *(zeige-dauer t)*, die die Darstellung analog zu *(zeige-7segment t)* übernimmt.

Falls Sie die 2er Gruppen der Segmente trennen möchten, können Sie folgende Grafik hierzu verwenden:

```
(underlay/xy  (underlay/xy (rectangle 100 200 'solid 'black)
                40 70 (rectangle 20 20 'solid 'red))
              40 140 (rectangle 20 20 'solid 'red))
```

2.5 Zusatzaufgabe: Timer

5 Zusatzpnkt.

Erstellen Sie anhand von Aufgabe 2.4 eine Funktion *(zeige-timer seconds t)*, die neben der Anzahl der Ticks t eine herunterzuzählende Zeit seconds (in Sekunden) erhält, und die die noch verbleibenden Sekunden grafisch darstellt. Falls die Zeit abgelaufen ist, geben Sie eine grafische oder textuelle Meldung aus. Ausprobieren können Sie ihre Funktion z.B. mit einem Countdown von zwei Minuten mit *(animate (curry zeige-timer 120))*.

Erreichbare Punkte: 35

Erreichbare Zusatzunkte: 5