Übungen zu Softwareentwicklung III, Funktionale Programmierung

Blatt 7, Woche 8

Leonie Dreschler-Fischer

WS 2015/2016

Ausgabe: Freitag, 27.11.2015,

Abgabe der Lösungen: bis Montag, 7.12.2015, 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 Generieren von Listen

Bearbeitungszeit 1 Std., 10 Pnkt.

Definieren Sie eine Funktion *range*, die ein Paar aus zwei Zahlen *interval* $'(start\ .\ stop)$ sowie eine Anzahl n als Argument nimmt und eine Liste errechnet, die den angebenden Bereich mit einer n-elementigen Liste beschreibt. hierbei ist *start* inklusive, *stop* ist hingegen nicht enthalten. Beispiel:

```
> (range '(0 . 10) 5) ⟶ '(0 2 4 6 8)
```

Programmieren Sie diese Funktion in drei Varianten:

- als allgemein rekursive Funktion,
- als endrekursive Funktion,
- mittels geeigneter Funktionen höherer Ordnung (*Tipp: build-list*).

2 Funktions-Plotter

Bearbeitungszeit 4 Std., ingesamt 25 Punkte

In dieser Aufgabe sollen Sie die Darstellung von Funktionen mittels eines Funktionen-Plotters implementieren. Diese Art von Anzeigen wird häufig in den unterschiedlichsten Wissenschaften eingesetzt, um Daten oder Funktionen besser "sichtbar" zu machen.

Verwenden Sie nach Möglichkeit Funktionen höherer Ordnung!

2.1 Funktionen und Werte

3 Pnkt.

Um Funktionswerte darstellen zu können, müssen Sie diese zunächst in einem zwei-dimensionalen Raum generieren. Verwenden Sie als Darstellung eines jeden Punktes Paare. Schreiben Sie eine Funktion, die für eine beliebige Eingabe-Funktion f sowie ein Intervall *interval* und eine Abtastung n dieses Intervalls erhält. Als Resultat sollte eine Liste der berechneten Punkte innerhalb des Intervalls erfolgen. Beispiel:

```
> (function->points sqr '(0 . 10) 5)
      ⟶ '( (0 . 0) (2 . 4) (4 . 16) (6 . 36) (8 . 64))
```

2.2 Wertebereiche

8 Pnkt.

Damit ein Funktionsresultat grafisch ansprechend dargestellt werden kann, müssen die in der vorigen Aufgabe ermittelten Funktionsstellen und Funktionswerte in andere Wertebereiche skaliert werden. Schreiben Sie zuerst eine Funktion die dies für eine Dimension vollzieht. Hierbei sollte den die alten auf die neuen Intervallgrenzen abgebildet werden und die Werte dazwischen interpoliert werden. Beispiel:

```
> (rescale1d '(0 2 4 6 8) '(10 . 50)) ⟶ '(10 20 30 40 50)
```

Schreiben Sie anschrießend eine zweite Funktion *rescale2d*, die unter Verwendung der 1D-Variante und zwei unterschiedlichen Ziel-Wertebereichen eine zweidimensionale Eskalierung erlaubt. Beispiel:

```
> (rescale2d '((0 . 0) (2 . 4) (4 . 16) (6 . 36) (8 . 64))
             '(10 . 50)
             '(5 . 25))
⟶ '((10 . 5) (20 . 27/4) (30 . 10) (40 . 37/4) (50 . 25))
```

2.3 Grafische Darstellung 1

4 Pnkt.

Schreiben Sie unter Zuhilfenahme des image-Paketes *(require 2htdp/image)* eine Funktion *(draw-points pointlist)*, die eine Liste von Punkten als blaue Ellipsen der Größe 1x1 auf einer leeren Szene grafisch darstellt. Verwenden Sie der Einfachheit halber eine Szenengröße von 800x600 Bildpunkten.

2.4 Grafische Darstellung 2

6 Pnkt.

Kombinieren Sie die bisher definierten Funktionen zu einer Funktion *plot-function*, die eine beliebige Funktion f in einem beliebigen Intervall *interval* unter Verwendung von n Stützstellen plotten kann. Testen Sie Ihre Funktion mit einigen Funktionen und Intervallen, und stellen Sie die Korrektheit der Darstellung sicher.

2.5 Simulation: Oszillograph

4 Pnkt.

Schreiben Sie ausgehend von Ihrer bisherige Funktion "plot-function" eine neue Variante *live-plot-function* so, dass sie zusätzlich einen weiteren Parameter t annimmt. Dieser Parameter soll die Markierung eines beliebigen Punkts des Plots erlauben, beispielsweise als rote Ellipse der Größe 5x5 Pixel. Falls der Parameter größer als die Liste der Punkt ist, soll wieder von vorne begonnen werden.

Verwenden Sie nun currying um die *live-plot-function* an eine bestimmte Funktion, einen Wertebereich sowie eine Menge von Stützstellen zu binden. Mit dem Animate-Framework *(require 2htdp/universe)* sollten Sie nun einen einfachen Oszillographen (mit 28 Ticks pro Sekunde) simulieren können:

```
(animate (curry live-plot-function ...) )
```

2.6 Zusatzaufgabe: Liniendarstellung

5 Zusatzpnkt.

Schreiben Sie eine alternative Form des Plotting-Verfahrens, das abweichend von der Aufgabe 2.3 keine Darstellung der Daten als Punkte erzeugt, sondern jeweils zwei Punkte durch eine Linie verbindet. Nennen sie diese Funktion *draw-lines* und erweitern Sie anschließend ihre beiden Plotting-Funktionen um einen weiteren Parameter, mit dem sich die Liniendarstellung aktivieren lässt.

Erreichbare Punkte: 35

Erreichbare Zusatzunkte: 5