Übungen zu Softwareentwicklung III, Funktionale Programmierung

Blatt 6, Woche 7

Leonie Dreschler-Fischer

WS 2017/2018

Ausgabe: Freitag, 1.12.2017,

Abgabe der Lösungen: bis Montag, 11.12.2017, 12:00 Uhr per email bei den Übungsgruppenleitern.

Ziel: Rekursion: Die Aufgaben auf diesem Zettel dienen dazu, sich mit dem Entwurf von rekursiven Funktionen vertraut zu machen. Sie entwerfen linear rekursive Funktionen und üben die unterschiedlichen Formen von Rekursion zu unterscheiden.

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 Formen der Rekursion

(Bearbeitungszeit 1 Std.)

Gegeben seien die folgenden Funktionsdefinitionen:

```
(define (take n xs)
 ;; das Kopfst\"uck einer Liste: die ersten n Elemente
 (cond
  ((null? xs) '())
  ((= 0 n) '())
  (else (cons (car xs)
              (take (- n 1) (cdr xs))))))

(define (drop n xs)
 ;; das Endst\"uck einer Liste entfernen
 (cond
  ((null? xs) '())
  ((= 0 n) xs)
  (else (drop (- n 1) (cdr xs)))))

(define (merge rel<? xs ys)
 ;; mische zwei vorsortierte Listen xs und ys
 ;; entsprechend der Ordnungsrelation rel<?
 (cond ((null? xs) ys)
   ((null? ys) xs)
    (else
      (let ((cx (car xs))
            (cy (car ys)))
           (if (rel<? cx cy)
             (cons cx (merge rel<? (cdr xs) ys))
             (cons cy (merge rel<? (cdr ys) xs)))))))

(define (merge-sort rel<? xs)
 ;; Sortiere eine Liste xs entsprechend
 ;; der Ordnungsrelation rel<?
  (let ((n (length xs)))
    (if (<= n 1) xs
      (let*((n/2 (quotient n 2))
            (part1 (take n/2 xs))
            (part2 (drop n/2 xs)))
           (merge
              rel<?
              (merge-sort rel<? part1)
              (merge-sort rel<? part2))))))
```

1. Welcher Typ von Rekursion liegt jeweils vor? (Mehrfachnennungen sind möglich) **Begründen** Sie Ihre Entscheidung. 10 Pnkt.

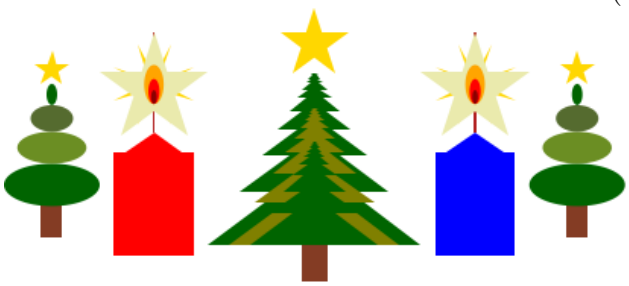
2. Welche Funktionen sind Funktionen höherer Ordung? Warum? 3 Pnkt.

3. Wandeln Sie die Funktion “take“ in eine endrekursive Funktion um. 3 Pnkt.

	lineare Rekursion	Baum-Rekursion	geschachtelte Rekursion	direkte Rekursion	indirekte Rekursion	End-rekursion
`take`	*ja/nein*	*ja/nein*	*ja/nein*	*ja/nein*	*ja/nein*	*ja/nein*
`drop`	*ja/nein*	*ja/nein*	*ja/nein*	*ja/nein*	*ja/nein*	*ja/nein*
`merge`	*ja/nein*	*ja/nein*	*ja/nein*	*ja/nein*	*ja/nein*	*ja/nein*
`merge-sort`	*ja/nein*	*ja/nein*	*ja/nein*	*ja/nein*	*ja/nein*	*ja/nein*

2 Ihre Nikolausaufgabe

(Bearbeitungszeit 4 Std.), 20 Pnkt.



Machen Sie sich mit dem Racket-Modul `"2htdp/image"` vertraut (siehe DrRacket-Hilfezentrum im Help-Menü).

In dieser Aufgabe ist Ihre Kreativität gefragt! Verwenden Sie grafische Elemente, wie Kreis, Rechteck usw., um ein festliches, weihnachtliches Bild zu komponieren, beispielsweise mit einem geschmückten Tannenbaum, Kerzen, Stapeln von Geschenken usw. Das Programm soll modular aufgebaut sein und gut kommentiert werden.

Ein Beispiel: Die kleinen Bäumchen rechts und links wurden aus olivgrünen Ellipsen, einem gelben Stern und einem braunen Rechteck zusammengesetzt, alles mit `above/align` zentriert übereinandergestapelt:

```
(define baum1 (above/align
               "center"
               ;; der Stern an der Spitze
               (star-polygon 40 5 2 "solid" "gold")
               ;; die Zweige
               (ellipse 20 40 "solid" "darkgreen")
               (ellipse 80 50 "solid" "darkolivegreen")
```

```
                (ellipse 130 60 "solid" "olivedrab")
                (ellipse 180 80 "solid" "darkgreen")
                ;; der Stamm
                (rectangle 40 60 "solid" "brown") ))
```

Das Bild soll wiederholte Elemente enthalten, die *rekursiv oder mit Funktionen höherer Ordnung* zu erzeugen sind. Wenn ihr Bild auch baumrekursive Strukturen enthält, können Sie sich noch Zusatzpunkte verdienen. 5 Zusatzpnkt.

```
(define UFO
  (underlay/align "center"
                  "center"
                  (circle 10 "solid" "green")
                  (rectangle 40 4 "solid" "green")))

(define UFO-BEAM
  (above/align "center"
               (underlay/align "center" "center"
                               (circle 10 "solid" "green")
                               (rectangle 40 4 "solid" "green"))
               (isosceles-triangle 100 20 "solid" "yellow")))

(define (create-UFO-scene t)
  (let ((height (* (sin (/ t 30)) 50)))
    (underlay/xy (rectangle 500 100 "solid" "white")
                 (modulo t 500) height
                 (if (< 0 height) UFO UFO-BEAM))))

(animate create-UFO-scene)
```

Weitere Zusatzpunkte gibt es, wenn Sie Ihr Weihnachtsbild mit einer Animation "aufpeppen". Um eine Animation zu erstellen binden Sie das Packet `"2htdp/universe"` ein. Anschließend können Sie eine Szene erstellen, die sich in Abhängigkeit von einem Argument (t) verändert. Das Argument t steht für "time" und gibt den Zeitpunkt in Sekunden, seit der Beginn der Animation an. Das Listing zeigt ein modifiziertes Beispiel aus dem Manual von DrRacket. Die Definition zweier geometrischer Objekte geschieht durch die Funktionen `UFO` und `UFO-BEAM`. Mit Hilfe von `create-UFO-scene` wird eine Animation der Objekte erstellt, die über die Funktion `animate` aufgerufen wird. 5 Zusatzpnkt.

Erreichbare Punkte: 36

Erreichbare Zusatzunkte: 10