Übungen zu Softwareentwicklung III, Funktionale Programmierung

Blatt 6, Woche 7

Leonie Dreschler-Fischer

WS 2012/2013

Ausgabe: Freitag, 30.11.2012,

Abgabe der Lösungen: bis Montag, 10.12.2012, 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 4 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 1/2 Std.), 10 Pnkt.

Gegeben seien die folgenden Funktionsdefinitionen:

```
(define (take n xs)
 ;; das Kopfstueck 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 Endstueck einer Liste
 (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))))))
```

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

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

2 Ihre Nikolausaufgabe:

(Bearbeitungszeit 3 1/2 Std.)

20 Pnkt.

Machen Sie sich mit dem Racket-Modul "image.ss" vertraut (siehe DrRacket-Hilfezentrum im Help-Menü). Laden Sie das Modul mit (**require** 2htdp/image) und verwenden Sie die grafischen Elemente, wie Kreis, Rechteck usw., um ein festliches, weihnachtliches Bild zu komponieren, beispielsweise mit einem geschmückten Tannenbaum, Kerzen, Stapeln von Geschenken usw.

Das Bild soll wiederholte Elemente enthalten, die rekursiv zu erzeugen sind; das Programm soll modular aufgebaut sein und gut kommentiert werden.

Ein Beispiel: Die kleinen Bäumchen rechts und links wurden aus olivgrü-

nen 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")
               ))
```

Wenn ihr Bild auch baumrekursive Strukturen enthält, können Sie sich noch Zusatzpunkte verdienen. 8 Zusatzpnkt.

Erreichbare Punkte: 30

Erreichbare Zusatzunkte: 8