SoSe 2014

Prof. Dr. Margarita Esponda

ProInformatik II: Funktionale Programmierung

1. Übungsblatt fürs Wochenende

Abgabe: 28. Juli 2012 um 8:55

Abgabe: Montag pünktlich um 8:55 Uhr vor Vorlesungsbeginn in **Papierform**.

1. Aufgabe (4 Punkte)

In der Welt der Bildverarbeitung werden unterschiedliche Formate für die Darstellung von Farben verwendet.

Es gibt z.B. das RGB-Format, in dem mit Hilfe von drei ganzen Zahlen zwischen 0 und 255 (Rot, Grün und Blau-Werte) die Farben kodiert werden.

In Zeitschriften und Büchern wird das CMYK-Format verwendet, in dem Bilder aus einer Kombination der Farben Cyan, Magenta, Yellow und Black (CMYK) gedruckt werden. Der Wertebereich der einzelnen Farbkomponenten liegt zwischen den Werten 0,0 und 1,0.

Schreiben Sie eine Haskell-Funktion mit dem Namen rgb2cmyk, die unter Verwendung folgender Formel nach Eingabe eines RGB-Tupels ein entsprechendes CMYK-Tupel berechnet.

Wenn RGB = (0, 0, 0), dann CMYK = (0.0, 0.0, 0.0, 1.0); sonst werden die Werte wie folgt berechnet:

w = max (R / 255, G / 255, B / 255)

C = (w − (R / 255)) / w

M = (w − (G / 255)) / w

Y = (w − (B / 255)) / w

K = 1− w

2. Aufgabe (2 Punkte)

Definieren Sie eine Funktion **minNeighborsDistance**, die den kleinsten Abstand zwischen zwei benachbarten Zahlen einer Liste findet.

Anwendungsbeispiel:

minNeighborsDistance [2, 3, 6, 2, 0, 1, 9, 8] => 1

3. Aufgabe (5 Punkte)

Eine Annäherung der Zahl π kann mit Hilfe folgender Reihe wie folgt berechnet

werden:

$$R_n = 4\sum_{k=0}^{n}\frac{(-1)^k}{2k+1} = 4\cdot(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-...) = \pi$$

Schreiben Sie eine Haskell-Funktion, die mit Hilfe einer rekursiven Funktion die Zahl π bei Eingabe von **n** annähert.

4. Aufgabe (3 Punkte)

Schreiben Sie eine Funktion **echtTeiler**, die bei Eingabe einer natürlichen Zahl **n** die Liste aller Teiler von **n** berechnet. Die Zahl **n** ist kein echter Teiler von **n** und soll deswegen nicht in der Ergebnisliste vorkommen.

Anwendungsbeispiel:

echtTeiler 250 => [1, 2, 5, 10, 25, 50, 125]

5. Aufgabe (2 Punkte)

Zwei natürliche Zahlen (m, n) werden als "Befreundetes Zahlenpaar" bezeichnet, wenn jede Zahl gleich der Summe der echten Teiler der anderen Zahl ist. Schreiben Sie eine Funktion, die bei Eingabe zweier natürlicher Zahlen entscheidet, ob die Zahlen befreundet sind oder nicht.

6. Aufgabe (10 Punkte)

Die Collatz-Folge ist eine interessante Folge, die im Jahr 1937 von Lothar Collatz entdeckt worden ist. Die Folge startet immer mit einer natürlichen Zahl $C_o = n \text{ mit } n > 0$ und berechnet alle weiteren Elemente C_i der Folge ab $i > 0$ mit Hilfe folgender Definition:

$$C_i = \begin{cases} \frac{C_{i-1}}{2} & \text{wenn } C_{i-1} \text{ gerade} \\ C_{i-1} \cdot 3 + 1 & \text{wenn } C_{i-1} \text{ ungerade} \end{cases}$$

Die Vermutung ist, dass egal mit welcher natürlichen Zahl n>0 gestartet wird, die Folge immer mit dem Zahlenzyklus 4, 2, 1 endet. Eine Vermutung, die bis jetzt noch nicht bewiesen worden ist.

a) Definieren Sie zuerst eine **nextCollatz** Hilfsfunktion, die bei Eingabe einer natürlichen positiven Zahl C_i die nächste Zahl C_{i+1} berechnet.

b) Definieren Sie eine **collatzSeq** Funktion, die bei Eingabe einer beliebigen natürlichen Zahl **n** die Collatz-Folge berechnet. Die Berechnung soll gestoppt werden, wenn die Zahl **1** erreicht wird. Das Ergebnis soll in einer Liste zurückgegeben werden.

Anwendungsbeispiel: **collatzSeq** 10 => [10, 5, 16, 8, 4, 2, 1]

c) Programmieren Sie eine **collatzSeqs** Funktion, die bei Eingabe einer natürlichen Zahl **n** die Collatz-Folge aller natürlichen Zahlen zwischen **1** und **n** berechnet und das Ergebnis als Liste von Listen zurückgibt.

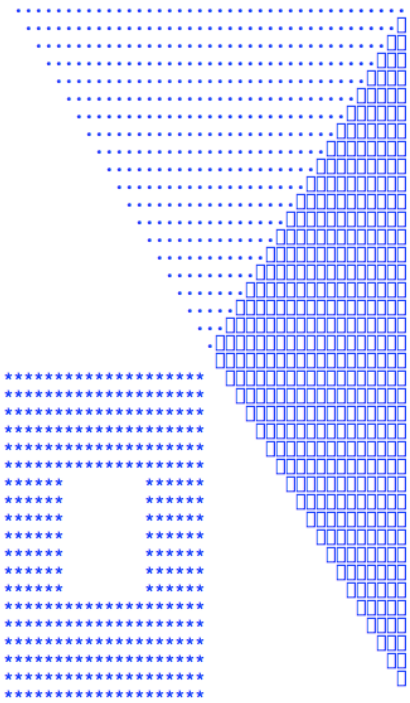
Anwendungsbeispiel:

collatzSeqs 5 => [[1], [2, 1], [3, 10, 5, 16, 8, 4, 2, 1], [4, 2, 1], [5, 16, 8, 4, 2, 1]]

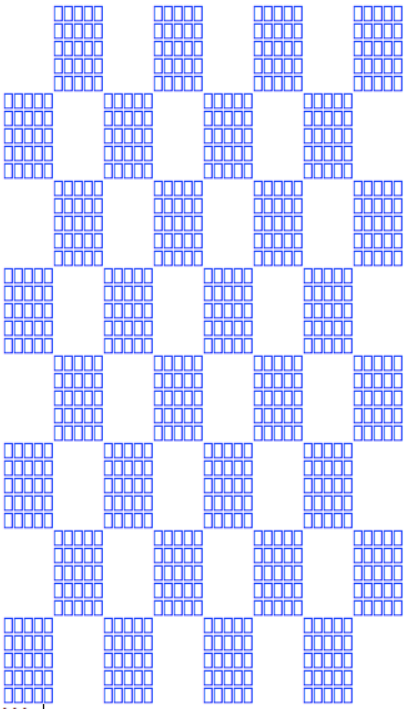
7. Aufgabe (6 Punkte)

In dieser Aufgabe sollen die Funktionen **square, easterEgg** und **chessboard** definiert werden, die die unten stehenden Zeichenbilder ausgeben.

square



chessboard



easteregg

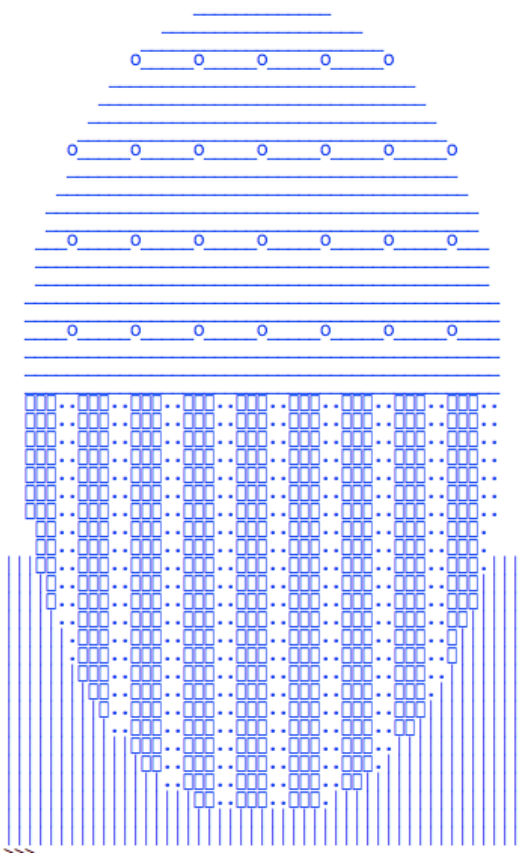


Bild deiner Wahl für Bonuspunkte

Eine **paintPicture** Funktion, die als Argument einen dieser Funktionsnamen bekommt, wird vorgegeben (siehe im Sakai). Diese Funktion darf nicht verändert werden. Die drei zu implementierenden Funktionen bekommen jeweils als Argumente ein (**x, y**, **size**) Tupel, das aus den **x, y** Koordinaten und **size**, der Seitenlänge des Bildes, besteht, und entscheiden, welches Zeichen an einer bestimmten Position zurückgegeben wird.

Innerhalb der zu implementierenden Funktionen darf keine Rekursion verwendet werden.

Die **printPicture** Funktion, so wie zwei Beispielfunktionen sind auf der Veranstaltungsseite unter Ressourcen herunter zu laden.

Wichtige Hinweise:

1) Verwenden Sie geeignete Namen für Ihre Variablen und Funktionsnamen, die den semantischen Inhalt der Variablen oder die Semantik der Funktionen wiedergeben.
2) Verwenden Sie vorgegebene Funktionsnamen, falls diese angegeben werden.
3) Kommentieren Sie Ihr Programme.
4) Verwenden Sie geeignete lokale Funktionen und Hilfsfunktionen in Ihren Funktionsdefinitionen.
5) Schreiben Sie in allen Funktionen die entsprechende Signatur.