Informatik A, WS 2016/17 — 7. Übungsblatt

Abgabe bis Freitag, 2. Dezember 2016, 12:00 Uhr, in die Fächer der Tutor/inn/en

38. Konjunktive Normalform, Programmieraufgabe, 13 Punkte

    (a) (2 Punkte) Definieren Sie einen rekursiven Datentyp `BoolAusdruck` für Boolesche Ausdrücke, analog zu den arithmetischen Ausdrücken aus der Vorlesung. Sie müssen damit zumindest Konstanten, Negation, Konjunktion und Disjunktion darstellen können, wahlweise auch die übrigen Junktoren.

    (b) (2 P.) Schreiben Sie dazu eine erweiterte Funktion `rechneAus` wie in Aufgabe 32.

    (c) (2 Punkte) Schreiben Sie eine Funktion `drucke` zum Anzeigen eines Booleschen Ausdrucks. Für die Junktoren können Sie `[nicht, und, oder, impl, aequiv, antiv, nand, nor] = "\172\8743\8744\8658\8660\8853\8892\8893"` oder zum Beispiel `[nicht,und,oder] = "~&|"` oder Haskell-Notation verwenden.

    (d) (6 Punkte) Schreiben Sie eine Funktion `knf`, die zu einer dreistelligen Booleschen Funktion die konjunktive Normalform bestimmt. Die Variablen sollen `"x1"`, `"x2"`, `"x3"` heißen.

    (e) (1 Punkt) Schreiben Sie eine Funktion `main:: IO()`, die eine kleine Testsuite mit einer Handvoll Beispielen durchrechnet und die Ergebnisse ausgibt.

39. QuickCheck, 0 Punkte

    Testen Sie ihr Programm für Aufgabe 38 mit QuickCheck auf folgende Eigenschaft:

```
prop_knf:: BF3 -> Bool -> Bool -> Bool -> Bool
prop_knf (BF3 h) x1 x2 x3 =
    rechneAus (knf h) [("x1",x1),("x2",x2),("x3",x3)] == h x1 x2 x3
```

    mit dem eingepackten Datentyp `BF3` für dreistellige Boolesche Funktionen.

```
data BF3 = BF3 (Bool -> Bool -> Bool -> Bool)
instance Arbitrary BF3 where
    arbitrary = do h <- arbitrary; return (BF3 h)
instance Show BF3 where show (BF3 x) = ...
```

    Für die `show`-Funktion können Sie die Lösung von Aufgabe 14 verwenden.

40. Listendurchlauf, Programmieraufgabe, 9 Punkte

    Erstellen Sie eine Tabelle für das kleine Einmaleins (die Produkte aller Zahlenpaare zwischen 1 und 10) als Zeichenkette. Die einzelnen Zeilen sind mit `'\n'` abgeschlossen. Das reduzierte Beispiel rechts zeigt, wie so eine Tabelle aussehen könnte, wenn man sie mit `putStr` ausgibt. Richten Sie die Spalten der Tabelle *vertikal aus*.

```
   |  1  2  10
---+----------
 1 |  1  2  10
 2 |  2  4  20
10 | 10 20 100
```

41. *ASCII art*, Programmieraufgabe, 8 Punkte

    Das folgende Programm `erzeugeBild` produziert, wenn man es mit einer Funktion `f::MalFunktion` füttert, eine aus Zeichen zusammengesetzte quadratische Graphik der Größe $g \times g$. Die Funktion f nimmt als Parameter die die Größe g des Bildes und die Spalten- und Zeilennummer (x, y), $1 \leq x, y \leq g$, und liefert das Zeichen, das an der Stelle (x, y) stehen soll.

```
type MalFunktion = Int -> (Int, Int) -> Char

erzeugeBild :: MalFunktion -> Int -> [Char]
erzeugeBild f größe = male größe [f größe (x,y) | y <- [1..größe],
                                                  x <- [1..größe]]
  where male 0 []      = []
        male 0 (c:cs)  = '\n' : (male größe (c:cs))
        male n (c:cs)  = c: (male (n-1) cs)
  -- Nach jeweis "größe" Zeichen wird ein Zeilenende eingefügt.
```

Die folgende Funktion erzeugt zum Beispiel das nachstehende Bild mit der Raute:

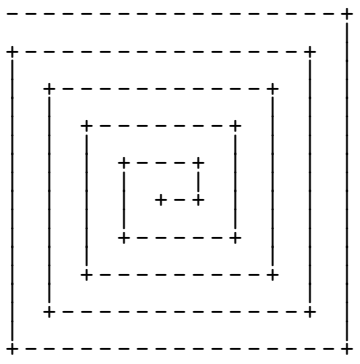
```
putStrLn (erzeugeBild raute 19)

raute :: MalFunktion
raute größe (x,y) =
   if x+y>halb && x+y<3*halb && abs (y-x)<halb then '|' else ' '
   where halb = (größe+1) `div` 2
```

Schreiben Sie entsprechende Funktionen für die beiden Bilder (a) Bauernhof und (b) Schachbrett. Das Ergebnis soll mit dem Größenparameter g skalieren. Für die Beispielgrößen ($g = 19$ und $g = 32$) müssen Sie die Beispielbilder reproduzieren.

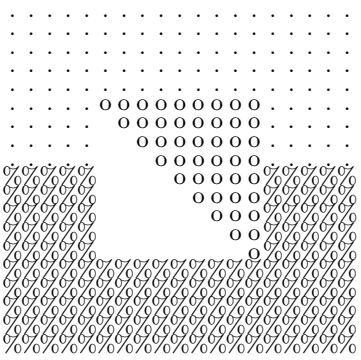
42. Datentypen, 0 Punkte

Betrachten Sie die Funktion `gruppiere` aus der Vorlesung vom Freitag, den 18. November.[1] Bestimmen Sie für jeden Teilausdruck den Datentyp.

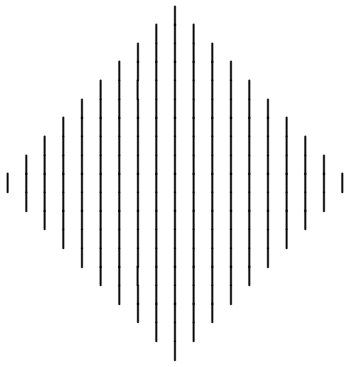
Spirale



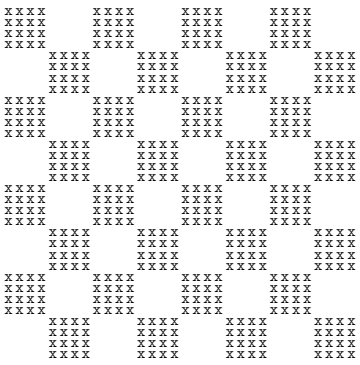
Bauernhof



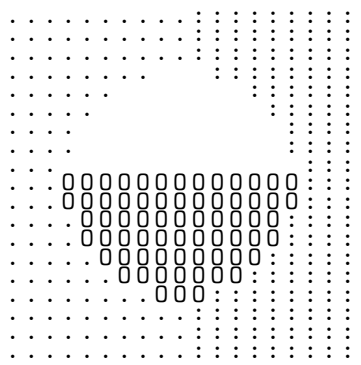
Raute



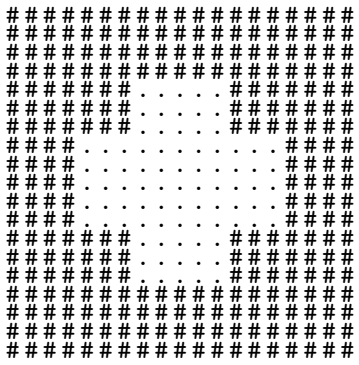
Schachbrett



Kreis



Schweizer Fahne



[1] `http://www.inf.fu-berlin.de/lehre/WS16/INFA/gruppen.hs`