Informatik A, WS 2016/17 — 1. Übungsblatt

Vorübungen, ohne Abgabe und ohne Bewertung.

---

1. Erste Schritte in Haskell

   (a) Laden Sie den *Glasgow Haskell Compiler* herunter und installieren Sie ihn auf Ihrem Rechner.

   (b) Starten Sie das Programm `ghci` und geben Sie die folgenden Befehle ein. Was passiert? Geben Sie jeweils eine kurze Interpretation.

   i. `8 + 10`
   ii. `10^20`
   iii. `3/4-2/3`
   iv. `1/7+1/7+1/7+1/7+1/7+1/7+1/7-1`
   v. `:?`
   vi. `3 + 5 * 7 == 40 - 2`
   vii. `putStr "Inf A\nWS16/17\n"`
   viii. `2^17 < 3^12`
   ix. `(2^10 <= 3^7) || (2^10 >= 3^7)`
   x. `(not True) || (False && True)`
   xi. `if 40 - 2 <= 5 then "Ja" else "Nein"`
   xii. `'B' < 'a'`
   xiii. `"Ferch" <= "Fahrt"`
   xiv. `"Ferch" <= "Fährte"`
   xv. `True < False`
   xvi. `2 * (4 +`
   xvii. `even 15`
   xviii. `:q`

   (c) Laden Sie die Datei `mystery.hs` von der Veranstaltungswebsite herunter und laden Sie sie mit `ghci`. Was passiert, wenn Sie `mystery 1 7 8`, `mystery 3 2 1` und `mystery 1 5 2` eingeben?
   Führen Sie weitere Experimente durch und fromulieren Sie eine Vermutung, was die Funktion `mystery` tut.

2. Logik im Alltag

   (a) Franks Frittenbude wirbt mit dem Slogan „Gutes Essen ist nicht billig!“. Die benachbarte Klopsmanufaktur Klaus kontert: „Billiges Essen ist nicht gut!“. Meinen sie nun dasselbe oder nicht?

   (b) Helmuts Herrenmagazin veröffentlicht folgende Ernährungstipps: „Wenn man zu einer Mahlzeit kein Bier trinkt, dann esse man Fisch. Wenn man Bier und Fisch zu einer Mahlzeit hat, dann verzichte man auf Eiscreme. Wenn man Eiscreme hat oder Bier meidet, dann esse man keinen Fisch.“ Der Gürkchenliebhaber Wolfgang moniert, das könne man auch prägnanter formulieren. Wie?

   (c) In Fahrplänen findet man oft Angaben wie „Mo–Fr wenn Werktag“, „an Samstagen, Sonntagen, und Feiertagen“, oder „an Werktagen außer Samstag“. Stellen Sie diese Bedingungen symbolisch mit Hilfe der Elementaraussagen (Mo) „Es ist Montag“, (Di), ..., (So), und (W) „Es ist ein Werktag“ und geeigneten Junktoren dar.

3. Es sei (A) „Sie ist alt“ und (B) „Sie ist weise“. Schreiben Sie in symbolischer Form

   (a) Sie ist alt und weise.

   (b) Sie ist weder alt noch weise.

   (c) Es stimmt nicht, dass sie jung oder weise ist.

   (d) Wenn sie alt ist, ist sie weise.

4. Beispiel für ein Expertensystem

   Meier, Schmid und Weber sind Pilot, Kopilot und Steward eines Flugzeugs, allerdings nicht unbedingt in der genannten Reihenfolge. Im Flugzeug befinden sich außerdem drei Reisende mit denselben drei Nachnamen. Um sie von der Besatzung zu unterscheiden, erhalten sie im folgenden ein „Herr“ vor ihre Namen. Wir wissen:

   (A) Herr Weber wohnt in München.

   (B) Der Kopilot wohnt in Hamburg.

   (C) Herr Schmid hat bereits vor langer Zeit seine Schulkenntnisse der Mathematik vergessen.

   (D) Der Fluggast, der denselben Nachnamen wie der Kopilot hat, lebt in Berlin.

   (E) Der Kopilot und einer der Passagiere, ein Mathematikprofessor, wohnen im gleichen Ort.

   (F) Meier spielt mit dem Steward regeläßig Tennis.

   Wie heißt der Pilot?

5. Knobelaufgabe

   Stellen Sie die Zahl 6 durch einen Ausdruck dar, der genau dreimal die Ziffer i enthält und ansonsten neben Klammern nur die 4 Grundrechenarten, Quadratwurzeln und die Fakultätsfunktion benutzen darf. Für $i = 2$ ist zum Beispiel $6 = 2 + 2 + 2$ oder $6 = 2 \times 2 + 2$. Lösen Sie die Aufgabe für alle $0 \leq i \leq 9$.

   (Diese Knobelaufgabe hat mehr mit Informatik zu tun, als man denkt: Was kann man mit beschränkten syntaktischen Mitteln ausdrücken?)

6. Abwiegen

   Sie besitzen 12 Ein-Euro-Münzen. Eine Münze ist gefälscht hat deshalb ein anderes Gewicht als die echten Münzen. Ihre Aufgabe ist es, mittels einer Balkenwaage die falsche Münze zu finden. Außerdem sollen Sie feststellen, ob sie leichter oder schwerer ist als eine echte Münze.

   (a) Sie dürfen nur jeweils eine Münze gegen eine andere abwägen. Beschreiben Sie ein Verfahren, das möglichst wenige Vergleiche verwendet. Wie viele sind das im schlechtesten Fall, und warum geht es nicht besser?

   (b) Sie dürfen mehrere Münzen gegeneinander wägen. Jetzt reichen drei Vergleiche aus! Beschreiben Sie ein solches Verfahren.

   Hinweis: Beginnen Sie mit vier Münzen auf jeder Seite. Der schwierige Fall tritt bei Ungleichheit ein. Dann gibt es noch acht Verdächtige.

   (c) Warum kann man nicht mit weniger als drei Vergleichen im schlimmsten Fall auskommen?

Informatik A, WS 2016/17 — 2. Übungsblatt

Abgabe bis Freitag, 28. Oktober 2016, 12:00 Uhr, schriftlich in die Fächer der Tutor/inn/en und Aufgabe 9b zusätzlich elektronisch im KVV.

---

7. (0 Punkte) Welche der folgenden Ausdrücke sind Aussagen?
   (a) Wien ist die Hauptstadt von Deutschland.
   (b) Ist n gerade, dann ist $n + 2$ gerade.
   (c) $x > y$.

8. If-then-else, 10 Punkte
   (a) Erstellen Sie die Wahrheitstafel für die dreistellige Boolesche Funktion, die durch das folgende HASKELL-Programm gegeben ist:

```
ifthenelse:: Bool -> Bool -> Bool -> Bool
ifthenelse a b c = if a then b else c
```

   (b) Schreiben Sie einen Booleschen Ausdruck für diese Funktion. (Bemühen Sie sich, einen möglichst einfachen Ausdruck zu finden.)

9. Zählen, 10 Punkte
   (a) (5 Punkte) Schreiben Sie einen Booleschen Ausdrucke in vier Variablen p, q, r, s, der genau dann wahr ist, wenn genau zwei dieser Variablen wahr sind.
   (b) (Programmieraufgabe, 5 Punkte) Setzen Sie die Lösung in eine HASKELL-Funktion

```
genau2:: Bool -> Bool -> Bool -> Bool -> Bool
```

   um. Schreiben Sie die Funktionsdefinition in eine Datei `genau2.hs` und laden Sie die Datei elektronisch im KVV hoch. Geben Sie zusätzlich das ausgedruckten Programm ab. Die erste Zeile der Datei muss die obige Typdeklaration sein.
   (c) (Freiwillige Zusatzaufgabe, 0 Punkte) Schreiben Sie eine möglichst einfach Funktion, die dieselben Ergebnisse wie `genau2` berechnet, wobei Sie sich nicht auf die Booleschen Operationen beschränken müssen.

10. Aussagenlogische Formulierung von Suchproblemen, 10 Punkte

    A(nton), B(erta), C(hristian) und D(orothea) fahren mit dem Zug und haben Platzkarten für ein Viererabteil, wobei die Plätze 1 und 2 in Fahrtrichtung blicken, die Plätze 3 und 4 rückwärts zur Fahrtrichtung liegen, und 1 und 3 Fensterplätze sind. Folgende Wünsche sind zu berücksichtigen:

    1) D will nicht rückwärts fahren,
    2) B und D wollen sich nicht gegenüber sitzen,
    3) A und B wollen nebeneinander sitzen,
    4) C wünscht einen Fensterplatz.

    (a) Überlegen Sie sich geeignete Boolesche Variablen zur Modellierung dieses Problems. Formulieren Sie die Bedingungen 1) bis 4) als Boolesche Ausdrücke.
    (b) Finden Sie eine Platzverteilung, die alle Wünsche erfüllt. Ist sie eindeutig?
    (c) (Zusatzfrage, 0 Punkte) Gibt es eine eindeutige Belegung der Variablen, die die Bedingungen Ihrer Lösung von Aufgabe (a) erüllt? Wenn nicht, formulieren Sie Boolesche Ausdrücke für zusätzliche Bedingungen, um Eindeutigkeit zu erzwingen.

Informatik A, WS 2016/17 — 3. Übungsblatt

Abgabe bis Freitag, 4. November 2016, 12:00 Uhr. Aufgabe 13b korrigiert am 28.10.

11. (0 Punkte) Stellen Sie die Wahrheitstafeln für folgende Ausdrücke auf.

    (a) $a \wedge \neg b$

    (b) $(a \wedge \neg b) \vee (\neg a \wedge b)$

    (c) $(a \vee \neg b) \Rightarrow ((a \wedge b) \vee (\neg a \wedge \neg b))$

    (d) $\neg\neg((a \wedge (b \vee b)) \wedge (a \wedge \neg\neg a))$

12. (0 P.) Welche der folgenden Aussagen sind Tautologien, welche sind Kontradiktionen?

    (a) $(a \vee \neg a) \wedge (b \vee \neg b)$

    (b) $(a \wedge \neg a) \wedge (a \Leftrightarrow b)$

    (c) $(p \wedge \neg p) \wedge (((q \vee \neg q) \Rightarrow p) \Leftrightarrow q)$

    (d) $[(a \vee b) \wedge (a \Rightarrow c) \wedge (b \Rightarrow c)] \Rightarrow c$

13. Umformung, 10 Punkte

    Beweisen Sie folgende Äquivalenzen, entweder durch Anwendung von Umformungsregeln oder durch Vergleichen der Wahrheitstafeln:

    (a) $p \Rightarrow (q \wedge r) \equiv (p \Rightarrow q) \wedge (p \Rightarrow r)$

    (b) $(p \vee s) \Rightarrow q \equiv (p \Rightarrow q) \wedge (s \Rightarrow q)$

14. Wahrheitstafel, 10 Punkte

    Schreiben Sie ein Haskell-Programm

    ```
    wTafel3:: (Bool->Bool->Bool->Bool) -> String
    ```

    das die Wahrheitstafel einer dreistelligen Booleschen Funktion erzeugt, im gleichen Format wie das Programm aus der Vorlesung.

    Für diese und auch alle zukünftigen Programmieraufgaben gelten die folgenden allgemeinen Hinweise, die auch auf der Netzseite stehen:

    - Laden Sie das Programm im KVV hoch und geben Sie es zusätzlich schriftlich ab.
    - Geben Sie bei allen Funktionen eine Typdeklaration an.
    - Verwenden Sie sprechende Namen, die den Inhalt der Variablen oder die Funktionalität der Funktion charakterisieren.
    - Verwenden Sie die vorgegebenen Funktionsnamen, falls diese angegeben sind.
    - Kommentieren Sie das Programm.
    - Verwenden Sie geeignete Hilfsfunktionen.
    - Löschen Sie Programmteile, die nicht verwendet werden.

15. KNF und DNF, 10 Punkte

    Bringen Sie $(p \wedge \neg q) \wedge ((q \vee \neg q) \Rightarrow p)$ auf (a) kanonische konjunktive Normalform und (b) auf kanonische disjunktive Normalform.

16. (0 Punkte) Stellen Sie fest, welche den angegebenen Bedingungen in den folgenden Formulierungen notwendige und welche hinreichende Bedingungen sind:

    (a) Wenn es regnet, wird die Straße nass.

    (b) Endet eine ganze Zahl auf 5 oder 0, so ist sie durch 5 teilbar.

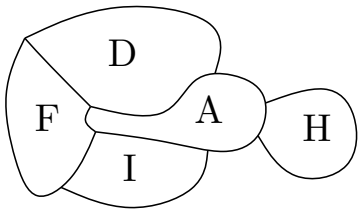
    (c) Für alle rellen Zahlen x gilt: Ist $x > 0$, so ist auch $x^2 > 0$.

    Gibt es darunter auch Bedingungen, die notwendige *und* hinreichende Bedingungen sind?

Informatik A, WS 2016/17 — 4. Übungsblatt

Abgabe bis Freitag, 11. November 2016, 12:00 Uhr. Aufgabe 17a klargestellt am 10.11.

17. Färben einer Landkarte, 10 Punkte



Die nebenstehende vereinfachte Landkarte von Europa soll mit drei Farben so gefärbt werden, dass Länder mit einer gemeinsamen Grenze verschiedene Farben haben.

(a) Schreiben Sie unter Verwendung geeigneter Boolescher Variablen eine Boolesche Formel, die die gültigen Färbungen beschreibt. Erklären Sie, was jede Variable bedeutet. Wieviele Variablen hat Ihre Formel? (Wenn Ihre Formel einem Schema folgt, können Sie ihn nach einigen Beispielen mit „usw." abkürzen, statt den ganzen Ausdruck explizit anzugeben.)

(b) Wie (a), aber Ihre Formel soll in konjunktiver Normalform sein. Wieviele Klauseln und Variablen hat Ihre Formel?

(c) Wieviele Möglichkeiten muss man betrachten, wenn man die Formel von Teil (b) durch Erstellen der Wahrheitstafel auf Erfüllbarkeit testen möchte?

(d) Wie lange würde das ungefähr dauern, wenn das Überprüfen einer Möglichkeit im Computer 0.02 Mikrosekunden dauert?

18. Programmieraufgabe, 10 Punkte.

Schreiben Sie eine Funktion `kleinerAlsDurchschnitt`, die berechnet, wie viele Zahlen in einer Liste von `Integer`-Werten *kleiner* als der Durchschnittswert sind. Bestimmen Sie selbst die richtige Signatur. Achten Sie auf die korrekte Behandlung der leeren Liste.

19. Preisberechnung, Programmieraufgabe, 10 Punkte

Listen der beiden folgenden Datentypen

```
type Einkaufsliste = [(String, Float)]
type Preisliste    = [(String, Float)]
```

geben an, *was* gekauft werden soll und *vieviel* (in einer passenden Einheit, z. B. kg), zum Beispiel `[("Mehl",0.5), ("Butter",0.25)]`, und andererseits den Preis in Euro pro Einheit für jeden Artikel.

(a) Definieren Sie eine Funktion

```
preis:: Preisliste -> Einkaufsliste -> (Float,[String])
```

zur Berechnung des Gesamtpreises aller Artikel einer Einkaufsliste. Die zweite Komponente des Ergebnisses soll die Liste der Artikelnamen enthalten, die in der Preisliste nicht gefunden wurden.

(b) Zusatzaufgabe, 0 Punkte. Erweitern Sie die Funktion so, dass der Preis für jeden gekauften Artikel der Einkaufsliste auf Cent gerundet wird. (Die Haskell-Funktion `round` rundet eine Zahl zur nächsten ganzen Zahl.)

20. Die de-Morgan'schen Gesetze, 0 Punkte: $\overline{p \wedge q} \equiv \bar{p} \vee \bar{q}$, $\overline{p \vee q} \equiv \bar{p} \wedge \bar{q}$

Leiten Sie das zweite Gesetz aus dem ersten her, indem Sie im ersten Gesetz (i) für p und q die Ausdrücke $\bar{x}$ und $\bar{y}$ einsetzen, (ii) beide Seiten negieren, (iii) doppelte Negationen elimineren und (iv) Variablen umbenennen. Funktioniert dieser Vorgang auch in die umgekehrte Richtung?

21. Äquivalenz und Antivalenz, 0 Punkte

    (a) Beweisen Sie, dass die Junktoren $\oplus$ und $\Leftrightarrow$ kommutativ und assoziativ sind.

    (b) Beweisen Sie: $\bar{x} \oplus y \equiv \overline{x \oplus y}$, $x \oplus \bar{y} \equiv \overline{x \oplus y}$. (Verneinung wandert auf die höhere Ebene.)

    (c) Beweisen Sie: $x \oplus x \equiv \mathit{falsch}$, $x \oplus \mathit{falsch} \equiv x$, $\bar{x} \equiv x \oplus \mathit{wahr}$

22. NAND, 0 Punkte

    Ist der Junktor $\bar{\wedge}$ assoziativ? Geben Sie einen Beweis oder ein Gegenbeispiel.

23. Zweierpotenzen und Zweierlogarithmen, 0 Punkte

    Genauso wie die Zweierpotenzen spielt auch die Umkehrfunktion, der Logarithmus zur Basis 2, in der Informatik eine große Rolle:

    $$b = \log_2 x \iff x = 2^b$$

    (a) Bestimmen Sie ohne Benutzung eines Taschenrechners die erste Dezimalziffer von $\log_2 32$, $\log_2 100$, $\log_2 1000$, und $\log_2 0.001$.

    (b) Wieviele Bits benötigt man, um ein Symbol abzuspeichern, das 26 verschiedene Werte annehmen kann?

    (c) Wieviele Bits benötigt man für n verschiedene Werte? Schreiben Sie eine Formel für die Anzahl der notwendigen Bits auf.

    (d) Schreiben Sie eine HASKELL-Funktion für Ihre Antwort aus (c). Den Logarithmus zu einer beliebigen Basis kann man mit der Funktion `logBase` bestimmen.

24. Geometrie, 0 Punkte

    Ein Kreis in der Ebenen kann als ein Tupel von 3 Gleitkommazahlen dargestellt werden. Die ersten beiden werden interpretiert als die x-y-Koordinaten des Mittelpunktes, die dritte als Radius. Schreiben Sie ein Haskell-Programm, das die folgenden Funktionen implementiert.

    (a) Die Funktion `istKreis` überprüft, ob die drei Zahlen tatsächlich einen Kreis mit nichtleerer Fläche definieren.

    (b) Die Funktion `flächeBBox` berechnet die Fläche der sogenannten *bounding box* zweier Kreise, also des kleinsten achsenparallelen Rechtecks, das beide enthält.

    (c) Die Funktion `istEnthalten` überprüft von zwei Kreisen, ob einer davon im anderen enthalten ist. (Eine Möglichkeit ist, den Abstand der Mittelpunkte zu verwenden.)

25. Bedingte Ausdrücke, 0 Punkte

    Schreiben Sie die folgende Funktion als einen *einzigen* nicht geschachtelten if-then-else-Ausdruck:

```
test:: Int->Int->Int->Bool
test x y z
  | x <= y    = True
  | y <= z    = False
  | otherwise = x < z
```

    Geht es auch ganz ohne if-then-else-Ausdruck und ohne Wächterbedingungen?

    (Die einfachste mir bekannte Lösung passt auf eine Zeile mit 15 Zeichen.)

Informatik A, WS 2016/17 — 5. Übungsblatt

Abgabe bis Freitag, 18. November 2016, 12:00 Uhr, in die Fächer der Tutor/inn/en

26. Konjunktive und disjunktive Normalform, 10 Punkte

    (a) Wandeln Sie den Ausdruck $(x_1 \vee x_2) \wedge (x_3 \vee x_4)$ in disjunktive Normalform um.

    (b) Wandeln Sie $(x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge (x_5 \vee x_6)$ in disjunktive Normalform um.

    (c) Wieviele Klauseln enthält die *kanonische* disjunktive Normalform von $(x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge (x_5 \vee x_6) \wedge (x_7 \vee x_8)$?

    (d) (Fortgeschrittene Zusatzfrage, 0 Punkte) Was ist die kleinstmögliche Anzahl an Klauseln für eine disjunktive Normalform des Ausdrucks in (b)?

    (e) Ist es eine gute Idee, einen logischen Ausdruck in disjunktive Normalform zu verwandeln, um ihn auf Erfüllbarkeit zu testen? Begründen Sie Ihre Antwort.

27. Resolutionskalkül, 0 Punkte

    (a) Ist $(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \neg x_3$ erfüllbar?

    (b) Ist $\bar{x} \vee (w \wedge \bar{x}) \vee (x \wedge \bar{w}) \vee (y \wedge z \wedge x) \vee (x \wedge \bar{y} \wedge z) \vee (w \wedge \bar{z} \wedge x)$ eine Tautologie?

    (c) Ist $(x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \neg x_3) \vee (\neg x_1 \wedge x_3) \vee \neg x_2$ eine Tautologie?

    (d) Zeigen Sie mit Resolution, dass $x_1 \wedge x_2 \wedge x_3$ eine Folgerung aus der Klauselmenge $\{\{\bar{x}_1, x_2\}, \{\bar{x}_2, x_3\}, \{x_1, \bar{x}_3\}, \{x_1, x_2, x_3\}\}$ ist.

28. Doppelresolution, 10 Punkte

    Man könnte die Resolution beschleunigen, indem man zwei Variablen gleichzeitig eliminiert: Aus den beiden Klauseln $\{x, y\} \cup K_1$ und $\{\bar{x}, \bar{y}\} \cup K_2$ bildet man den „Doppelresolventen“ $K_1 \cup K_2$ bezüglich der Variablen x und y.

    Beispiel: In der Klauselmenge

    $$M = \{\{p, q, r\}, \{\bar{p}, \bar{q}, s\}, \{p, t\}, \{\bar{t}, q\}, \{\bar{q}, \bar{u}\}, \{u, \bar{p}\}, \{\bar{p}, \bar{r}\}, \{\bar{q}, \bar{s}\}, \{\bar{r}, s\}, \{r, \bar{s}\}\}$$

    könnte man dann aus den ersten beiden Klauseln p und q eliminieren und die Klausel $\{r, s\}$ herleiten.

    Zeigen Sie, dass M erfüllbar, aber $M \cup \{\{r, s\}\}$ ein Widerspruch ist. Die „abgekürzte Resolution“ ist daher *nicht gültig*.

29. Teilworttest, 0 Punkte

    Implementieren Sie eine Funktion `istTeilwort:: Eq a => ([a],[a])->Bool` zum Testen, ob die erste Eingabeliste als Teilwort von aufeinanderfolgenden Elementen der zweiten auftritt. Sie können die Funktion `beginntMit` aus der Vorlesung verwenden.

30. Unterfolgentest, Programmieraufgabe, 10 Punkte

    Implementieren Sie eine Funktion `istUnterfolge:: Eq a => ([a], [a]) -> Bool` zum Testen, ob die erste Eingabeliste Unterfolge der zweiten ist. Im Gegensatz zu einem Teilwort müssen bei einer *Unterfolge* die Zeichen nur in der richtigen Reihenfolge auftreten, aber nicht direkt hintereinander. Also ist IFOA Unterfolge von INFORMATIK, aber kein Teilwort. ORMA ist ein Teilwort und somit auch Unterfolge.

31. Quotient und Rest, 0 Punkte

    Außer `div` und `mod` gibt es in Haskell auch die analogen Operationen `quot` und `rem`, die bei negativen Operanden andere Ergebnisse liefern. Was passiert mit der Funktion `ziffern` aus der Vorlesung, wenn man diese anderen Operationen einsetzt?

Informatik A, WS 2016/17 — 6. Übungsblatt

Abgabe bis Freitag, 25. November 2016, 12:00 Uhr, in die Fächer der Tutor/inn/en

32. Resolutionskalkül, 0 Punkte

    (a) Wie viele verschiedene mögliche Klauseln kann man mit n Variablen bilden?

    (b) Wie viele Klauseln können bei Anwendung des Resolutionskalküls maximal entstehen, wenn es n Variablen gibt und alle Klauseln die Länge 2 haben?

    (c) Sei nun $n \geq 3$. Geben Sie Beispiele für eine Menge von Klauseln der Länge 3, für die der Resolutionskalkül eine Klausel erzeugt, die alle n Variablen enthält.

33. Ausdrücke mit Variablen, Programmieraufgabe, 11 Punkte

    (a) Erweitern Sie die Funktion `rechneAus` aus der Vorlesung um einen zweiten Parameter vom Typ `[(String,Integer)]`, der eine Liste von Variablennamen mit zugehörigen Werten enthält. Beispiel:

    ```
    rechneAus ((3+Var "c")*(4+Var "b")) [("a",5),("b",-5),("c",3)]=-6
    ```

    Variablen, die nicht in der Liste vorkommen, sollen als 0 betrachtet werden.

    (b) Schreiben Sie dafür zum Bestimmen der Wertes einer Variablen eine polymorphe Hilfsfunktion `schaueNach:: Eq a => a -> [(a,b)] -> Maybe b`. Beispiele:
    `schaueNach "c" [("a",5),("b",-5),("c",3)] = Just 3`
    `schaueNach "d" [("a",5),("b",-5),("c",3)] = Nothing`

34. Textverarbeitung, Programmieraufgabe, 10 Punkte

    Schreiben Sie eine Funktion `ersetze:: String -> String -> String -> String`, sodass `ersetze alt neu s` das erste Vorkommen von `alt` in der Zeichenkette `s` durch `neu` ersetzt. Beispiel: `ersetze "ich" "du" "sicherlich" = "sduerlich"`. Wenn `alt` nicht in `s` vorkommt, soll das Ergebnis `s` sein.

35. Uhrzeit, Programmieraufgabe, 0 Punkte

    Machen Sie den Typ `data Uhrzeit = Zeit Int Int` aus der Vorlesung zu einem Exemplar der Typklasse `Show` mit einer eigenem Funktion `show`, die die Uhrzeit in der Form `"x:y Uhr"` ausgibt. Die Minuten sollen immer zweistellig erscheinen.

36. Klammern bei arithmetischen Ausdrücken, 0 Punkte

    (a) Die Funktion `drucke` aus der Vorlesung geht beim Anzeigen eines arithmetischen Ausdrucks auf Nummer sicher, indem sie um jeden Ausdruck eine Klammer setzt. Gibt es trotzdem die Möglichkeit, den Benutzer zu täuschen und einen Ausdruck zu schreiben, der nicht seine echte Struktur wiedergibt?

    (b) Schreiben Sie eine modifizierte Funktion `druckeKnapp:: Ausdruck -> String`, die alle überflüssigen Klammern weglässt, z.B. `1*2-3+4*5/(6*7-8)-9`

37. Vollkommene Zahlen, Listendurchlauf, Programmieraufgabe, 9 Punkte

    (a) Schreiben Sie eine Funktion `teiler`, die die Liste der positiven Teiler einer Zahl in irgendeiner Reihenfolge bestimmt. Beispiel: `teiler 45 = [45,3,15,1,9,5]`.

    (b) Bestimmen Sie jede Zahl n zwischen 1 und 1000, bei der die Summe ihrer von n verschiedenen Teiler größer ist als die Zahl n selbst.
    (Hinweis: Die Funktion `sum` berechnet die Summe der Elemente einer Liste)

    (c) Bestimmen Sie alle *vollkommenen Zahlen* zwischen 1 und 1000: die Zahlen n, die gleich der Summe ihrer von n verschiedenen Teiler sind.

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:

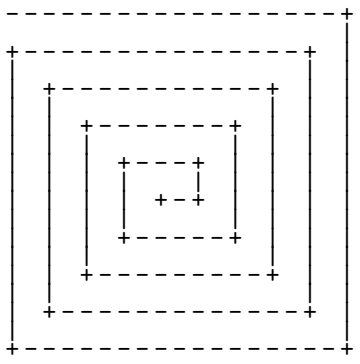
```
putStrLn (erzeugeBild raute 19)

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.

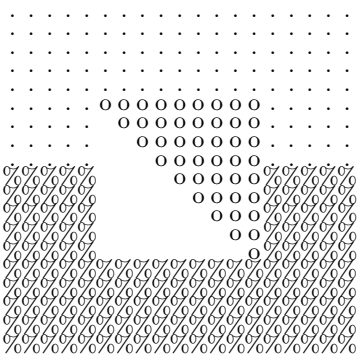
42. Datentypen, 0 Punkte

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

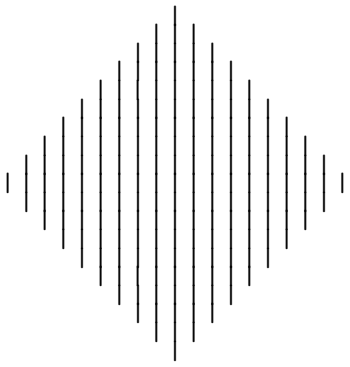
Spirale



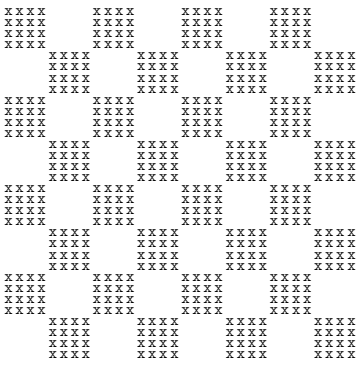
Bauernhof



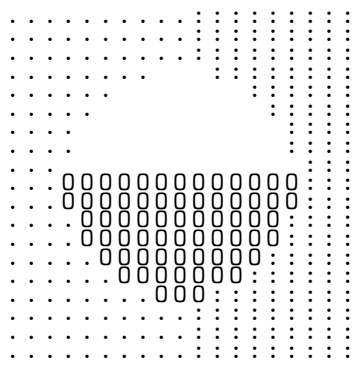
Raute



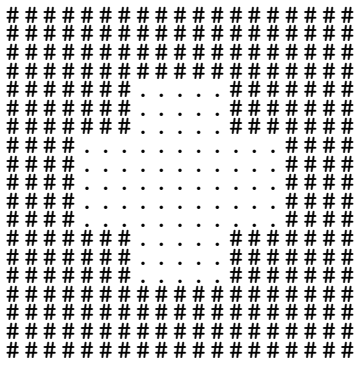
Schachbrett



Kreis



Schweizer Fahne



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

Informatik A, WS 2016/17 — 8. Übungsblatt

Abgabe bis Freitag, 9. Dezember 2016, 12:00 Uhr, in die Fächer der Tutor/inn/en

Schreiben Sie von nun an *für jede Programmieraufgabe* eine Funktion `main:: IO()`, die eine kleine Testsuite mit einer Handvoll Beispielen durchrechnet und die Ergebnisse ausgibt.

43. Inverse Funktionen, 10 Punkte

    Schauen Sie die Definitionen der Funktionen `lines` und `unlines` (und der benötigten Hilfsfunktionen) in der Spezifikation des HASKELL-Sprachstandards[1] nach. Sind die Funktionen invers zueinander? Für welche x gilt

    `lines (unlines x) == x`?

    Für welche x gilt

    `unlines (lines x) == x`?

    Sie können QuickCheck zu Hilfe nehmen.

44. Spiegeln und Drehen von zweidimensionalen Mustern (10 Punkte)

```
\      +-->
 \     |
  +---+
```

    In Aufgabe 41 haben wir Buchstabengrafiken erzeugt, die aus durch `'\n'` abgeschlossenen Zeilen bestehen und mit `putStr` ausgegeben werden. Zum Beispiel liefert `putStr "\\      +-->\n \\     |\n  +---+\n"` das obige Bild. (Verkehrte Schrägstriche müssen in der Zeichenkette doppelt eingegeben werden, damit man sie von Sonderkodierungen wie `\n` unterscheiden kann.)

    Schreiben Sie Funktionen `flipH`, `flipV`, und `flipD`, die eine solches Bild an einer horizontalen Achse, an einer vertikalen Achse, beziehungsweise an einer Diagonale von links oben nach rechts unten spiegeln. Schreiben Sie Funktionen `dreheL` und `dreheR`, die das Bild um 90° nach links beziehungsweise nach rechts drehen. Das Ergebnis soll bei dem Beispiel so aussehen:

```
flipH:   >--+     \   flipV:   +---+      flipD:  \    dreheL:  >
            |    \             \   |               \            -
            +----+              \  +-->             +           -
                                                    -           +|+
                                                    -             -
                                                    -             -
                                                  +|+             -
                                                  -               +
                                                  -                \
                                                  >                \
```

    Hinweis: Die Bibliotheksfunktion `reverse::[a]->[a]` kehrt die Reihenfolge einer Liste um. Manche Funktionen können aus anderen zusammengesetzt werden.

45. Faltung, 10 Punkte

    (a) Die Funktion `map` kann auch sehr knapp durch eine Faltung definiert werden:

    ```
    map2 :: (a->b) -> [a] -> [b]
    map2 h = foldr ((:).h) []
    ```

    Geben Sie für *jeden* Teilausdruck in dieser Definition (also zum Beispiel auch für „`foldr`“ und „`.`“) den Datentyp in Abhängigkeit von `a` und `b` an.

    (b) Programmieraufgabe. Definieren Sie die Funktionen `filter`, `any`, `unlines` und `takeWhile` (siehe Fußnote [1]) als linke oder rechte Faltung. Verwenden Sie dabei neue Namen `filter'`, `any'`, usw.

46. Hornklauseln, 0 Punkte

    Überprüfen Sie die folgende Boolesche Formel mit Resolution auf Erfüllbarkeit.

    $$(\neg a \vee \neg b \vee \neg d) \wedge (\neg e) \wedge (\neg c \vee a) \wedge (c) \wedge (b) \wedge (\neg g \vee d) \wedge (g)$$

[1] `https://www.haskell.org/onlinereport/haskell2010/haskellch9.html`

Informatik A, WS 2016/17 — 9. Übungsblatt

Abgabe bis Freitag, 16. Dezember 2016, 12:00 Uhr, in die Fächer der Tutor/inn/en

47. Hornklauseln, 10 Punkte

    (a) Wenden Sie den Markierungsalgorithmus auf die folgende Hornformel an:
    $$(\neg a \vee \neg b \vee \neg d) \wedge (\neg e) \wedge (\neg c \vee a) \wedge (c) \wedge (b) \wedge (\neg g \vee d) \wedge (g)$$
    (b) Wir haben Apparate zur Durchführung folgender chemischer Reaktionen:
    $$\begin{aligned} \mathrm{MgO} + \mathrm{H_2} &\to \mathrm{Mg} + \mathrm{H_2O}, \\ \mathrm{C} + \mathrm{O_2} &\to \mathrm{CO_2}, \\ \mathrm{H_2O} + \mathrm{CO_2} &\to \mathrm{H_2CO_3}. \end{aligned}$$
    Ferner sind in unserem Labor die Stoffe MgO, $\mathrm{H_2}$, $\mathrm{O_2}$ und C vorhanden. Man beweise mit Hilfe des Markierungsalgorithmus, dass man dann $\mathrm{H_2CO_3}$ herstellen kann.
    (c) Diskutieren Sie den Zusammenhang zwischen dem Markierungsalgorithmus und dem Resolutionskalkül.

48. Quantoren, 0 Punkte

    Bestimmen Sie die Wahrheitswerte der folgenden Aussagen und begründen Sie Ihre Antworten:

    (a) $\forall z \in \mathbb{Z}\colon \exists x \in \mathbb{Z}\colon \forall y \in \mathbb{Z}\colon xy = z$
    (b) $\exists x \in \mathbb{R}\colon \forall z \in \mathbb{R}\colon \exists y \in \mathbb{R}\colon xy = z$
    (c) $\forall x \in \mathbb{Z}\colon \exists z \in \mathbb{Z}\colon \forall y \in \mathbb{Z}\colon xy = z$
    (d) $\exists z \in \mathbb{R}\colon \forall x \in \mathbb{R}\colon \exists y \in \mathbb{R}\colon xy = z$

49. Formulieren Sie folgende Aussagen in prädikatenlogischer Sprache. (0 Punkte)

    (a) Das Quadrat jeder negativen ganzen Zahl ist negativ.
    (b) Alle ganzen Zahlen sind negativ oder alle ganzen Zahlen sind positiv.
    (c) Die Gleichung $x^2 = 1$ hat mindestens zwei Lösungen.
    (d) Die Gleichung $x^2 = 1$ hat höchstens zwei Lösungen.
    (e) Die Gleichung $x^2 = 1$ hat genau zwei Lösungen.
    (f) Das unten definierte Prädikat $\mathit{teilt}(a, b)$ besagt, dass a ein Teiler von b ist. (Man sagt auch, dass b durch a teilbar ist.)
    $$\mathit{teilt}(a, b) :\Leftrightarrow \exists m \in \mathbb{Z}\colon am = b$$
    Formulieren Sie unter Verwendung dieses Prädikats die Aussage: Eine ganze Zahl ist genau dann durch 6 teilbar, wenn sie durch 2 und durch 3 teilbar ist.
    (g) Jede ganze Zahl ist ein Teiler von 0.
    (h) Nicht jede ganze Zahl ist ein Teiler von 100.

50. Primzahlen, Prädikatenlogik, Negation, 0 Punkte

    (a) Eine Primzahl ist eine ganze Zahl größer als 1, die keine anderen Teiler außer 1 und sich selbst hat. Definieren Sie das entsprechende Prädikat $\mathit{prim}(n)$ in Prädikatenlogik. (Verwenden Sie das Prädikat „*teilt*“ aus der vorigen Aufgabe.)

(b) Die folgende Aussage, die schon bei Euklid bewiesen ist, besagt, dass es unendlich viele Primzahlen gibt:

$$\forall m \in \mathbf{Z}\colon \exists n \in \mathbf{Z}\colon n > m \wedge \mathit{prim}(n) \tag{1}$$

Schreiben Sie die Verneinung dieser Aussage. Formulieren Sie das Ergebnis so um, dass das Ergebnis kein Negationssymbol mehr enthält.

(c) Erläutern Sie, warum die Aussage (1) bedeutet, dass es unendlich viele Primzahlen gibt, obwohl dort die Zahl „unendlich“ gar nicht vorkommt.

(d) Formulieren Sie die Goldbachsche Vermutung von 1742, dass sich jede gerade Zahl > 2 als Summe von zwei Primzahlen schreiben lässt. (Diese Vermutung ist bis heute ungelöst.)

51. Überlappende Listen, 10 Punkte

Wir bezeichnen mit L_n die Menge der Listen $x = [x_0, x_1, \ldots, x_{n-1}]$ vom Typ `[Int]` mit Länge n. Wir sagen, dass zwei Listen $x, y \in L_n$ sich mit einem Anteil α überlappen, wenn sie eine gemeinsame Teilfolge der Länge mindestens αn haben:

$$\begin{aligned}\mathit{überlappen}(x, y, \alpha) :\Leftrightarrow \exists i, j, k, l \in \mathbb{Z}\colon (j - i = l - k) \wedge (0 \le i, j, k, l < n)\\ \wedge (j + 1 - i \ge \alpha n) \wedge \forall u \in \mathbb{Z}\colon (0 \le u \le j - i \to x_{i+u} = y_{k+u})\end{aligned}$$

(a) Eine Beispiel: Die Listen $x = [7, 9, 30, 20, 3, 1]$ und $y = [30, 20, 7, 1, 2, 4]$ überlappen sich zu einem Drittel. Was ist i, j, k, l, n, α in diesem Beispiel?

(b) Formulieren Sie unter Verwendung des Prädikats *überlappen* folgende Aussage in der Sprache der Prädikatenlogik:

> Für drei beliebige Listen $x, y, z \in L_n$ gilt: Wenn sich je zwei dieser Listen um *mehr als* 50 % überlappen, dann enthalten die Listen eine gemeinsame Zahl.

Ihre Formulierung soll keine freien Variablen enthalten. *Alle* Variablen müssen quantifiziert sein.

(c) Zusatzfrage, 0 Punkte: Ist die obige Aussage wahr?

52. Fahrplan, Programmieraufgabe, 10 Punkte

Man kann einen Fahrplan komprimieren, indem man aufeinanderfolgende Zeilen mit gleichem Inhalt zusammenfasst.[1] Der Wochentagsfahrplan der Linie 186[2]. könnte zum Beispiel so aussehen:

```
 00    | 18 38
01-03  | -
 04    | 58
 05    | 18 38 58
 06    | 18 28 38 48 58
07-20  | 08 18 28 38 48 58
21-23  | 18 38 58
```

Schreiben Sie ein Programm, dass einen Fahrplan in dieser Form ausgibt. „Leere Zeilen“ (wie im Beispiel zwischen ein und vier Uhr) sollen nicht ignoriert werden. Sie können Programme aus der Vorlesung und aus den bisherigen Übungen kombinieren und modifizieren, und Sie können Standardfunktionen wie `groupBy` verwenden.

[1] Der Straßenfahrplan von Graz `http://www.inf.fu-berlin.de/lehre/WS16/INFA/Fahrplan-Graz.pdf` zeigt diese Idee in einer anderen Darstellungsart.

[2] siehe `http://www.inf.fu-berlin.de/lehre/WS16/INFA/Fahrplan-Berlin.pdf`, Seite 1

Informatik A, WS 2016/17 — 10. Übungsblatt

Abgabe bis Freitag, 6. Januar 2017, 12:00 Uhr, in die Fächer der Tutor/inn/en

---

53. Keine Negation bei der Vorbedingung, 0 Punkte

    Die Aussage „Für alle x mit der Eigenschaft $P(x)$ gilt $Q(x)$“ lässt sich so formulieren:

    $$\forall x\colon P(x) \to Q(x)$$

    Zeigen Sie mit den de-Morganschen Regeln und weiteren Umformungen, dass die Negation dieser Aussage zu folgender Aussage äquivalent ist:

    $$\exists x\colon P(x) \wedge \neg Q(x)$$

    Die Bedingung $P(x)$ ist hier *nicht* negiert.

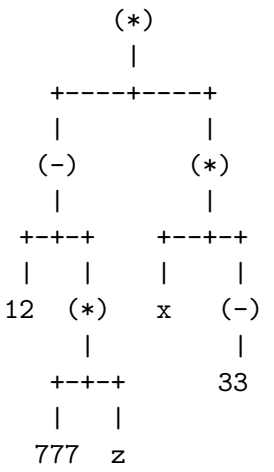
    Formulieren Sie die letzte Aussage in Worten.

54. Zeichnen von Weihnachtsbäumen, 0 Punkte

    Schreiben Sie ein Programm, das einen arithmetischen Ausdruck (Vorlesung vom 11. November 2016) mit Buchstabengrafik als Baum ausgibt. Der Ausdruck

    ```
    (12-777*Var "z")*(Var "x"*(-33))
    ```

    könnte zum Beispiel wie im nebenstehenden Bild aussehen. Sie dürfen eine Annahme über den Wertebereich der Daten in den Blättern machen, wenn Sie sie dokumentieren.



55. Wie viele gleiche? Programmieraufgabe, 10 Punkte

    (a) Schreiben Sie eine Funktion `wieVieleGleiche:: Eq a => a->a->a->a->Int`, die bestimmt, wieviele der vier Argumente gleich sind. Beispiel: `wieVieleGleiche 4 9 2 4 = 2`, weil das erste und vierte Argument gleich sind.

    (b) Diese Aufgabe ist nicht genügend klar und genau spezifiziert. Ergänzen Sie die Spezifikation. (Ihr Programm muss natürlich zur Spezifikation passen.)

56. Fibonacci-Paare, Programmieraufgabe, 10 Punkte

    (a) Schreiben Sie ein Programm `fib2:: Int -> (Integer,Integer)`, das bei Eingabe von n das Paar (F_{n-1}, F_n) aus zwei aufeinanderfolgenden Fibonaccizahlen ausrechnet. Dabei soll die Berechnung von `fib2 n` auf einen einzigen rekursiven Aufruf von `fib2 (n-1)` zurückgeführt werden.

    (b) Berechnen Sie damit die ersten 100 Fibonaccizahlen.

57. Vollständige Induktion, 0 Punkte

    Beweisen Sie folgende Aussagen mit vollständiger Induktion nach n.

    (a) Genau in der Hälfte aller Listen der Länge n vom Typ `[Bool]` ist eine *ungerade* Anzahl von Elementen gleich `True`. (Und: Für *welche* Werte von n gilt das?)

    (b) Für $a, r \in \mathbb{R}$ mit $r \neq 1$ und für alle $n \geq 0$ gilt: $\displaystyle\sum_{i=0}^{n} ar^i = \frac{ar^{n+1} - a}{r - 1}$.

58. Gruselfibonacci, 0 Punkte

    Schreiben Sie ein Programm, das die ersten 20 Glieder der Folge $G_n = G_{n-1} - G_{n-2}$ bei vorgegebenen Startwerten G_0 und G_1 ausgibt.

59. Wahrheitstafel, Programmieraufgabe mit Wettbewerb, 10 Punkte

    Schreiben Sie Haskell-Funktionen `wTafel1`, `wTafel2`,..., `wTafel6`, für Wahrheitstafeln mit bis zu sechsstelligen Booleschen Funktionen in Erweiterung der Funktion `wTafel''` aus der Vorlesung vom 26. 10. und der Aufgabe 14 vom 3. Übungsblatt.

    Wettbewerb. Die Lösung mit der kleinsten Anzahl an *Tokens* gewinnt! Tokens sind Bezeichner (Namen), Operatoren, Klammern, Zahlen und so weiter. Die Länge von Bezeichnern oder von Operatoren, die aus mehreren Symbolen bestehen, wie `<=` oder `++`, ist egal. Infixoperatoren wie `` `div` `` zählen als ein einziges Token. Typdeklarationen, die in separaten Zeile stehen, werden nicht mitgezählt, genauso wie Leerzeichen und Kommentare. Sie dürfen auch Sprachkonstrukte verwenden, die nicht in der Vorlesung behandelt wurden. Die *vollständige* Lösung (außer Funktionen aus den Standardbibliotheken) muss innerhalb von Kommentarzeilen `{-WETT-}` und `{-TTEW-}` stehen, um zu zählen. Zum Beispiel:

    ```
    {-WETT-}
    wTafel1  ::  (Bool -> Bool) -> String
    wTafel2  ::  (Bool -> Bool -> Bool) -> String
    ...
    wTafel1 f = ...
    {-TTEW-}
    ```

    Nennen Sie Ihre Datei `Wahrheitstafel.hs`. Wie immer müssen Sie eine `main`-Funktion zum Testen schreiben, aber diese kann außerhalb der `{-WETT-}`...`{-TTEW-}`-Klammern stehen.

    Die Einsender/innen der 10 besten Lösungen werden veröffentlicht, und die schönsten Lösungen werden ins Internet gestellt. Es gibt hier keine Bonuspunkte, dafür aber jede Menge Spaß und Ehre. Wenn Sie nicht möchten, dass Ihr Name im Internet erscheint, können Sie entweder auf den Wettbewerb verzichten, indem Sie die Kommentarzeilen `{-WETT-}`...`{-TTEW-}` weglassen, oder den Dozenten anschreiben. Für die normale Punktebewertung entsteht Ihnen dadurch kein Nachteil.

60. Generische Wahrheitstafel, 0 Punkte (schwierig)

    Schreiben Sie in Haskell eine *einzige* polymorphe Funktion `wTafel`, die die Wahrheitstafel für Boolesche Funktionen mit beliebiger Stellenanzahl ausgibt.

61. Definition von linker Faltung durch rechte Faltung, 0 Punkte (zum Tüfteln)

    ```
    foldl2 :: (a -> b -> a) -> a -> [b] -> a
    foldl2 op a0 bs = foldr step id bs a0
       where (b `step` gs) a = gs (a `op` b)
    ```

    (a) Welchen Typ hat `step` in Abhängigkeit von den Typvariablen `a` und `b`?

    (b) Zeigen Sie wie `foldl2 op` a_0 `[`b_1, b_2, b_3`]` = `(foldr step id [`b_1, b_2, b_3`])` a_0 = `(`b_1 `` `step` `` `(`b_2 `` `step` `` `(`b_3 `` `step` `` id)))` $a_0$ dasselbe Ergebnis liefert wie `foldl op` $a_0$ `[`$b_1, b_2, b_3$`]` = `((`a_0 `` `op` `` b_1`)` `` `op` `` b_2`)` `` `op` `` b_3

    (c) Definieren Sie auf analoge Art unter Verwendung von `foldl` eine Funktion `foldr2`, die das Gleiche macht wie `foldr`

Informatik A, WS 2016/17 — 11. Übungsblatt

Abgabe bis Freitag, 13. Januar 2017, 12:00 Uhr, in die Fächer der Tutor/inn/en

---

62. Prädikatenlogik, Belegungen, 10 Punkte

    Eine Funktion $b\colon U \to \{w, f\}$ heißt eine *Belegung* eines Booleschen Ausdrucks F, wenn U eine Variablenmenge ist, die alle Variablen enthält, die in F vorkommen.

    Für zwei Belegungen $b, b'\colon U \to \{w, f\}$ und eine Variable $x \in U$ ist das Prädikat $\mathit{diff}(b, b', x)$ folgendermaßen definiert:

    $$\mathit{diff}(b, b', x) :\Leftrightarrow \text{ } b \text{ und } b' \text{ unterscheiden sich nur in der Variablen } x.$$

    Formulieren Sie diese Definition als prädikatenlogische Formel. Verwenden Sie nur die Prädikate $\in, =, \neq, \subseteq$, und das dreistellige Prädikat $K(h, A, B)$, das besagt, dass h eine Funktion von A nach B ist. (Sie können für dieses Prädikat auch die konventionelle Schreibweise $h\colon A \to B$ verwenden.) Die Wahrheitswerte heißen w und f.

63. Fibonacci-Zahlen, vollständige Induktion, 10 Punkte

    (a) B_n sei die Anzahl der Additionen, die bei der Berechnung `fibo n` mit der ursprünglichen Funktion `fibo` aus der Vorlesung vom 14. Dezember 2016 ausgeführt werden.
    Schreiben Sie eine rekursive Definition für B_n.

    (b) Bestimmen Sie die Werte $B_0, B_1, \ldots, B_{10}$.

    (c) Formulieren Sie eine Vermutung, wie die Folge B_n mit den Fibonacci-Zahlen zusammenhängt.

    (d) Beweisen Sie diese Vermutung durch vollständige Induktion.

    (e) (Zusatzfrage, 0 Punkte) Geben Sie eine einfache Erklärung für den Zusammenhang.

64. Schaltkreise, 0 Punkte

    Konstruieren und zeichnen Sie einen logischen Schaltkreis für den Ausdruck $(A \wedge C) \vee (\neg B)$, der nur NAND-Elemente verwendet.

65. Programmieraufgabe: Gleitkommazahlen, Rundungsfehler, 10 Punkte

    Berechnen Sie die Summe

    $$\sum_{n=1}^{10^6} \frac{1}{n^2} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots + \frac{1}{(10^6)^2}$$

    (a) mit `Float`, (b) mit `Double`.

    (c) Wenn man n gegen unendlich gegen lässt, dann strebt die Summe gegen den Grenzwert $\sum_{n=1}^{\infty} \frac{1}{n^2} = \pi^2/6$ (die Eulersche Reihe). Wie weit sind die Ergebnisse aus (a) bzw. (b) vom exakten Grenzwert entfernt?

    (d) Berechnen Sie die Differenz zwischen den Ergebnissen aus (a) und (b).

    (e) Wiederholen Sie die gesamte Aufgabe mit 10^7 statt 10^6.

66. Strukturelle Induktion, 0 Punkte

    Beweisen Sie `map (f . g)` = `map f . map g`, indem Sie die Gleichung

    ```
    map (f . g) as = (map f . map g) as
    ```

    für alle Listen `as` durch strukturelle Induktion nach `as` beweisen.

Informatik A, WS 2016/17 — 12. Übungsblatt

Abgabe bis Freitag, 20. Januar 2017, 12:00 Uhr, in die Fächer der Tutor/inn/en

---

67. Natürliche Zahlen, Programmieraufgabe, 10 Punkte

    Man kann sich in HASKELL seine eigenen natürlichen Zahlen als algebraischen Datentyp definieren:

    ```
    data Nat = Zero | Succ Nat
    ```

    `Zero` steht für 0 und `Succ` für Nachfolger (*successor*). Die Zahl 3 wird zum Beispiel durch `Succ (Succ (Succ Zero))` dargestellt.

    (a) Schreiben Sie eine Funktion `natToInt:: Nat -> Int` zum Umwandeln von einer Darstellung in die andere.

    (b) Die Addition kann man folgendermaßen definieren:

    ```
    add :: Nat -> Nat -> Nat
    add Zero m = m                    -- add.1
    add (Succ n) m = Succ (add n m) -- add.2
    ```

    Beweisen Sie mit struktureller Induktion, dass diese Addition assoziativ ist: `(x ‘add‘ y) ‘add‘ z = x ‘add‘ (y ‘add‘ z)`. Überlegen Sie, welche Variable Sie als Induktionsvariable wählen. Begründen Sie jeden Schritt, indem Sie angeben, welche Definitionsgleichung (add.1 oder add.2) oder welche anderen Eigenschaften sie verwenden.

68. Entwurf eines Schaltnetzes, 10 Punkte

    Konstruieren Sie ein Schaltnetz mit drei Eingängen $x, y, z \in \{0, 1\}$ und zwei Ausgängen $a, b \in \{0, 1\}$. Der Ausgang a soll zu 0 ausgewertet werden, wenn y oder z gleich 1 ist. Der Ausgabe b wird zu 1 ausgewertet, falls die Zahl $x + y \cdot 2 + z \cdot 2^2$ durch 3 teilbar ist. Zeichnen Sie die Gatter entsprechend der DIN–Norm.

69. Endliche Automaten, 0 Punkte

    Konstruieren Sie einen endlichen Automaten mit Eingabealphabet $\text{In} = \{A, C, T\}$ und Ausgabealphabet $\text{Out} = \{0, 1\}$, der genau dann 1 ausgibt, wenn das bisherige Eingabewort die Folge $ACAT$ als Teilwort (siehe Aufgabe 29 vom 5. Blatt) enthält.

    Zeichnen Sie das Zustandsdiagramm.

70. Lauflängenkodierung (run-length encoding), Programmieraufgabe, 10 Punkte

    Bei der Lauflängenkodierung wird eine Zeichenkette `"aaaabbaaac"` mit vielen wiederholten Zeichen komprimiert, indem man die Länge jedes *Laufes* von gleichen Zeichen nimmt: `[(4,'a'), (2,'b'), (3,'a'), (1,'c')]`. Schreiben Sie eine Funktion `komp`, die diese Kodierung berechnet, und die Umkehrfunktion `dekomp` für die Dekodierung.

71. Nichtassoziative Faltung von Listen, 0 Punkte

    Beschreiben Sie das Ergebnis der Funktion

    ```
    differenzen:: Integer -> Integer -> Integer -> Integer
    differenzen a b c = foldr (-) a [b..c]
    ```

    durch eine explizite Formel in den Größen a, b, c (oder mehrere Formeln, die für verschiedene Fälle passen). Beweisen Sie Ihre Formel, zum Beispiel durch vollständige Induktion nach $c - b$, oder auch direkt.

Informatik A, WS 2016/17 — 13. Übungsblatt

Abgabe bis Freitag, 27. Januar 2017, 12:00 Uhr. Aufgabe 72 ergänzt am 19. Januar

72. Strukturelle Induktion, 10 Punkte

Beweisen Sie

```
elem x (a ++ b) = elem x a || elem x b
```
(1)

für endliche Listen a und b unter Verwendung folgender Definitionen:

```
elem :: (Eq a) => a -> [a] -> Bool
elem x []     = False              -- elem.1
elem x (y:ys) = x==y || elem x ys  -- elem.2

(++) :: [a] -> [a] -> [a]
[]     ++ ys  = ys                 -- (++.1)
(x:xs) ++ ys  = x : (xs ++ ys)     -- (++.2)

(||) :: Bool -> Bool -> Bool
True  || _ = True                  -- (||.1)
False || x = x                     -- (||.2)
```

Begründen Sie jeden Schritt, indem Sie angeben, welche Definitionsgleichung (zum Beispiele `(++.1)` oder `elem.2`) oder welche anderen Eigenschaften sie verwenden.

73. Strikte Funktionen, 10 Punkte

(a) Ist die Funktion `elem` aus der vorigen Aufgabe strikt im ersten Argument?

(b) Ist sie strikt im zweiten Argument?

(c) Gilt die Gleichung (1) auch für unendliche Listen a und/oder b? (Die Gleichung ist auch dann erfüllt, wenn auf beiden Seiten das Ergebnis $\bot$ herauskommt.)

(d) Gilt sie auch für Listen mit undefinierten Elementen?

Begründen Sie Ihre Antworten.

74. Träge Auswertung, 10 Punkte

Zeigen Sie Schritt für Schritt, wie Haskell den Wert `h 3 5` im Zusammenhang mit folgenden Definitionen auswertet:

```
h:: Int -> Int -> Int
h m n | nichtLeer xs = vorne xs
      | otherwise    = n
   where xs = ints m

ints m = m : ints (m+1)

vorne (x:y:zs) = x+y
vorne [x] = x

nichtLeer []    = False
nichtLeer (_:_) = True
```

75. Träge Multiplikation, Programmieraufgabe, 0 Punkte

Schreiben Sie eine *polymorphe* Funktion `mult` zur Multiplikation zweier Zahlen, die das zweite Argument nicht auswertet, wenn das erste Argument 0 ist.

76. Funktionsdefinitionen, 0 Punkte

(a) Welche der folgenden Definitionen bewirkt, dass die Funktion `g = entf`n `' '` alle Leerzeichen aus einer Zeichenkette entfernt?

```
entf1 z l      = [if x==z then "" else [x] | x <- l]
entf2 z l      = [x | x <- l, x/=z]
entf3 z        = concat . map (\ x -> if x==z then "" else x)
entf4 z []     = []
entf4 z (z:zs) = entf4 z zs
entf4 z (x:xs) = x: entf4 z xs
entf5 z        = foldr (\ x r -> if x==z then r else x:r) []
```

Bestimmen Sie für jede Funktion `entf`n, ob sie überhaupt nach den Haskell-Regeln gültig ist, und stellen Sie gegebenenfalls ihren Typ fest.
Begründen Sie Ihre Antworten.

(b) Was macht die Funktion

```
entf '.' . entf ',' . entf '.' . entf ';' . entf '.'
```

wenn `entf` die richtige Lösung zu Aufgabe (a) ist?

77. Sichtbarkeitsbereiche, 0 Punkte

(a) Geben Sie für jeden im folgenden Programmstück eingeführten Namen den Sichtbarkeitsbereich an.

```
h a b
  | r > 0     = a+r
  | otherwise = -a+b
   where
      g c a = a*b*c - 1
      r     = g a (-b)
```

(b) Benennen Sie die Variablen so um, dass jeder Name nur an einer einzigen Stelle definiert ist.

(c) Bestimmen Sie `[h (-1) (-1), h (-2) 1, h (-1) 2]`.

78. Strukturelle Induktion, 0 Punkte

Wie kann man `foldr g z (a ++ b)` durch `foldr` mit den Listen `a` und `b` ausdrücken?

Beweisen Sie Ihre Formel, indem Sie für `(++)` die Definition von Aufgabe 72 und folgende Definition für `foldr` verwenden:

```
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr f z []     =  z                    -- foldr.1
foldr f z (x:xs) =  f x (foldr f z xs)   -- foldr.1
```

79. Alle drin, Programmieraufgabe, 0 Punkte

192 ist die kleinste ganze 3-stellige Zahl n, so dass n, $2n$ und $3n$ gemeinsam alle Ziffern von 1 bis 9 enthalten. Berechnen Sie eine Liste `hasAllDigits :: [Int]` mit allen 3-stelligen Zahlen n, für die n, $2n$ und $3n$ alle Ziffern von 1 bis 9 enthalten. Sie können auch eine allgemeinere Funktionen schreiben, also zum Beispiel, dass in der Eingabe mit spezifiziert ist, was drin sein soll usw.

Informatik A, WS 2016/17 — 14. Übungsblatt

Abgabe bis Freitag, 3. Februar 2017, 12:00 Uhr in die Fächer der Tutor/inn/en

80. Teilsummen, 0 Punkte

(a) Die vordefinierte Funktion `scanl` ist ähnlich wie `foldl`, aber es wird eine Liste aller Zwischenergebnisse erzeugt:

```
scanl,scanl2 :: (a -> b -> a) -> a -> [b] -> [a]
scanl f q []     = [q]
scanl f q (x:xs) = q : scanl f (f q x) xs
```

(b) Was ist das Ergebnis von `scanl (+) 0 [3,5,-1,7]`?

(c) Weil auf der rechten Seite der Definition nur ein einziger rekursiver Aufruf mit einer um 1 kürzeren Liste steht, kann `scanl` mit einer Liste der Länge n in $O(n)$ Zeit berechnet werden, sofern die Funktion `f` in konstanter Zeit (in $O(1)$ Zeit) berechnet werden kann.
Wie lange dauert die Berechnung mit dem folgenden einfachen Programm?

```
scanl2 f q xs = [foldl f q (take i xs) | i <- [0..length xs]]
```

Geben Sie die Laufzeit mit O-Notation an.

(d) Ab welcher Listenlänge macht sich der Unterschied zwischen `scanl (+) 0` und `scanl2 (+) 0` in der Laufzeit bemerkbar, wenn man das Programm einmal in der Befehlszeile aufruft? Lassen Sie die Summe der Ergebnisliste ausrechnen, damit der Bildschirm nicht überschwemmt wird.

(e) Wie lange dauert die Berechnung von `length (scanl (+) 0 xs)` beziehungsweise `length (scanl2 (+) 0 xs)`, wenn `xs` eine Liste der Länge n ist?

81. Der Teilabschnitt mit der größten Summe, 10 Punkte

Die folgenden drei Funktionen `maxSum1, maxSum2, maxSum3 :: (Num a, Ord a) => [a] -> a` bestimmen für eine Liste $[a_1, \ldots, a_n]$ die größte Summe $a_i + a_{i+1} + \cdots + a_j$ einer in ihr enthaltenen *zusammenhängenden* Teilfolge.

```
maxSum1 list = maximum (0: [su i j | j <- [1..length(list)],
        i <- [1..j]])    -- direkt nach Definition
  where su a b = sum (drop (a-1) (take b list))
maxSum2 [] = 0 -- mit "scan"
maxSum2 (x:xs) = max (maxSumStart (x:xs)) (maxSum2 xs)
maxSumStart :: (Num a, Ord a) => [a] -> a
-- Teilfolgen, die am Anfang von xs beginnen:
maxSumStart xs = maximum (scanl (+) 0 xs)
maxSum3 = maxSumA 0  -- die größte Summe ist mindestens 0
maxSumA :: (Num a, Ord a) => a -> [a] -> a
maxSumA akk [] = akk -- ohne Erklärung. akk steht für "Akkumulator"
maxSumA akk (x:xs) | x <= -akk = max akk (maxSumA 0  xs)
                   | otherwise = max akk (maxSumA (akk+x) xs)
```

Bestimmen Sie eine obere Schranke für die Laufzeit für jede der drei Funktionen in Abhängigkeit von der Listenlänge n, in O-Notation. Begründen Sie Ihre Antworten.

82. Laufzeiten, 0 Punkte

Wir haben fünf Algorithmen, deren Laufzeit in Abhängigkeit von der Größe n der Eingabe auf einem bestimmten Rechner (1) $10n$, (2) n^3+5n, (3) $2^n/100$, (4) $3n^2+4n$, (5) $5n \log_2 n$ Mikrosekunden ist.

(a) Wie große Probleme kann man mit diesen Algorithmen (i) in einer Sekunde, (ii) in 1 Stunde, (iii) in 1 Woche lösen?

(b) Wie ändert sich die Antwort bei einem Rechner, der tausendmal so schnell ist?

83. Typklassen und Typinferenz, 0 Punkte

Bestimmen Sie die Typen der Standardfunktionen `curry f x y = f(x,y)`, `uncurry f (x,y) = f x y`, und `id x = x`. Stellen Sie fest, welche der folgenden zwölf Funktionen gültig sind, und bestimmen Sie gegebenfalls ihren Typ:

```
concat . map show                map . (++)
curry id                         uncurry curry
uncurry id                       curry uncurry
curry (curry id)                 uncurry uncurry
uncurry (uncurry id)             curry curry
uncurry (<=)                     uncurry . (==)
```

Verwenden Sie den Computer höchstens zum Überprüfen Ihrer Antworten. Füttern Sie jede gültige Funktion mit den erforderlichen Argumenten, sodass HASKELL ein Ergebnis ausgibt.

84. Typklassen und Typinferenz, 10 Punkte

Stellen Sie fest, welche der folgenden vier Funktionen gültig sind, und bestimmen Sie gegebenfalls ihren Typ, wobei `flip f x y = f y x` ist.

```
map reverse                      flip . curry
curry flip                       flip flip
```

Leiten Sie Schritt für Schritt her, wie sich der Typ aus den Restriktionen des Typsystems ergibt. Verwenden Sie den Computer höchstens zum Überprüfen der Antworten.

Gegeben Sie für jede gültige Funktion einen Beispielaufruf mit den nötigen zusätzlichen Argumenten an, für den HASKELL ein Ergebnis ausdruckt.

85. Ausgeglichene Klammern, Programmieraufgabe, 10 Punkte

Schreiben Sie eine Funktion `klammerTest`, die testet, ob die in einer Zeichenkette vorkommenden öffnenden und schließenden runden Klammern „`(`“ und „`)`“ einen korrekten Klammerausdruck bilden. Das bedeutet: in jedem Anfangsstück (Präfix) treten mindestens so viele öffnende wie schließende Klammern auf, und insgesamt gibt es ebenso viele öffnende wie schließende Klammern. Beispielsweise sind `()(())` und `((())()(()))` gültig, nicht aber `(()(` oder `)()()(`.

86. Induktion, 0 Punkte

Für welche Werte von k und l ist die Gleichung

```
(take k) . (take l) = take (min k l).
```

gültig? Beweisen Sie sie unter Verwendung folgender Definitionen:

```
take :: Int -> [a] -> [a]
take n _   | n <= 0 =  []                  -- take.0
take _ []           =  []                  -- take.1
take n (x:xs)       =  x : take (n-1) xs   -- take.2
min x y  | x <= y    =  x                  -- min.1
         | otherwise =  y                  -- min.2
```

Es bietet sich eine Fallunterscheidung an, je nachdem ob k oder l größer ist.

Informatik A, WS 2016/17 — 15. Übungsblatt

Bonuspunkte, freiwillige Abgabe bis Freitag, 10. Februar 2017, 12:00 Uhr

87. Wörterbuchoperationen in einem binärem Suchbaum, 10 Bonuspunkte

    (a) Fügen Sie die Schlüssel I, N, F, O, R, M, A, T, K, Z in dieser Reihenfolge in einen anfangs leeren binären Suchbaum ein. Zeichnen Sie den Baum nach jedem Einfügevorgang.

    (b) Wenn wir nun nach U suchen, mit welchen Schlüsseln wird U dabei verglichen?

    (c) Löschen Sie die Schlüssel Z, O, N. Zeichnen Sie wieder den Baum nach jedem Löschvorgang.

    (d) Angenommen, wir suchen den Schlüssel 363 in einem binären Suchbaum. Bestimmen Sie für jede der folgenden Schlüsselfolgen, ob es einen Suchbaum gibt, wo sie als Folge der Vergleichsschlüssel auf dem Suchpfad nach 363 auftreten kann. Begründen Sie jeweils Ihre Antwort.

        i. 2, 399, 387, 219, 266, 382, 381, 278, 363.

        ii. 2, 252, 401, 398, 330, 344, 397, 363.

        iii. 935, 278, 347, 621, 299, 392, 358, 363.

        iv. 925, 202, 911, 240, 912, 245, 363.

        v. 924, 220, 911, 244, 898, 258, 362, 363.

88. Binäre Suchbäume, Programmieraufgabe, 5 Bonuspunkte

    In der Vorlesung wurde der algebraische Datentyp `Suchbaum` definiert. Implementieren Sie folgende Funktionen.

    (a) (5 Punkte) Bestimme die Höhe eines Binärbaums.

    ```
    höhe:: Suchbaum s w -> Int
    ```

    Ein Baum mit einem einzigen Knoten hat die Höhe 0, und der Baum `Nil` soll die Höhe -1 haben.

    (b) (0 Punkte) Teste, ob ein gegebener Binärbaum die Suchbaumeigenschaft hat.

    ```
    istSuchbaum:: Ord s => Suchbaum s w -> Bool
    ```

    Variante i. Verwenden Sie die Funktion `toList` aus der Vorlesung.

    Variante ii. Kommen Sie ohne `toList` oder eine vergleichbare Funktion aus.

89. Binärbäume, 5 Bonuspunkte

    Beweisen Sie mit vollständiger Induktion: Jeder binäre Suchbaum mit n „echten Knoten“ (Datensätze mit dem Konstruktor `Knoten`) hat $n + 1$ `Nil`-Knoten.

90. Wörterbuchoperationen mit Listen, 0 Punkte

    Implementieren Sie die Wörterbuchoperationen durch Listen, analog zur Vorlesung.[1]

```
class Wörterbuch wb where
  einfügen :: ...
type ListenWörterbuch s w = [(s,w)]
instance Wörterbuch ListenWörterbuch where
  einfügen x wert ...
```

[1] `http://www.inf.fu-berlin.de/lehre/WS16/INFA/Woerterbuch.hs` zum Abspeichern (kodiert mit UTF-8) oder `Wooerterbuch.hs` zum Lesen im Browser.

91. Traversieren von binären Suchbäumen, Programmieraufgabe, 0 Punkte

Eine *Traversierung* eines binären Suchbaums T ist eine Funktion, welche alle Knoten von T systematisch besucht. Es gibt vier Arten der Traversierung:

- **Preorder**: Besuche erst die Wurzel, dann rekursiv den linken Teilbaum, gefolgt von dem rechten Teilbaum.
- **Inorder** (symmetrische Reihenfolge): Besuche erst rekursiv den linken Teilbaum, danach die Wurzel, dann den rechten Teilbaum.
- **Postorder**: Besuche erst rekursiv den linken Teilbaum, dann den rechten Teilbaum und schließlich die Wurzel.
- **Levelorder**: Besuche die Knoten nach aufsteigender Tiefe und alle Knoten mit der gleichen Tiefe von links nach rechts.

(a) Schreiben Sie Funktionen

```
preorder, inorder, postorder, levelorder:: Suchbaum s w -> [(s,w)]
```

die die Einträge eines Suchbaumes in der Reihenfolge, wie sie besucht werden, als Liste zurückgeben.

Versuchen Sie dabei, die `++`-Operation zu vermeiden und lineare Laufzeit zu erreichen.

(b) Die Preorder-Traversierung eines binären Suchbaums mit 10 Einträgen ergibt die folgende Schlüsselfolge: M B A F D P N O Z R. Die Postorder-Traversierung liefert A D F B O N R Z P M. Wie sieht der Baum aus?

92. Kodierung, 0 Punkte

(a) Welche der folgenden Binärcodes sind eindeutig dekodierbar? Welche sind präfixfrei? Warum?

$$C_1 = \{0110, 010, 0\}, \quad C_2 = \{010, 00, 001, 01\}, \quad C_3 = \{1, 001, 0100, 011\}.$$

(b) Konstruieren Sie für die folgenden Kodewortlängen einen binären präfixfreien Kode und zeichnen Sie den zugehörigen Kodebaum.

$$n_1 = 1,\ n_2 = 2,\ n_3 = 3,\ n_4 = n_5 = 5,\ n_6 = n_7 = n_8 = 6,\ n_9 = 7.$$

93. Kreuzsicherungscode, Programmieraufgabe, 10 Bonuspunkte

Der *Kreuzsicherungscode* wandelt ein 0-1-Wort der Länge n^2 in ein 0-1-Wort der Länge $n^2 + 2n$ um, indem er die Eingabe als $(n \times n)$-Matrix auffasst und für jede der n Zeilen und n Spalten jeweils ein Paritätsbit anhängt.

(a) Schreiben Sie eine Kodierungsfunktion `encode::Int -> [Int] -> [Int]` für den Kreuzsicherungscode. Dabei soll der erste Parameter die Dimension n der Code-Matrix angeben, und der zweite Parameter ist eine 0-1-Folge, deren Länge ein Vielfaches von n^2 ist.

(b) Schreiben Sie einer entsprechende Decodierfunktion `decode::Int -> [Int] -> [Int]`, die einen Fehler korrigieren kann: Für jede 0-1-Folge x passender Länge und jede Funktion `kippeEinBit::[Int] -> [Int]`, die die Identitätsfunktion ist oder die höchstens ein einzelnes Bit einer Bitfolge verändert, muss gelten:

```
(decode n . kippeEinBit . encode n) x = x
```

94. Alle verschieden, 0 Punkte

- Eingabe: Eine Liste `xs::Ord t => [t]`.
- Ausgabe: `True`, falls alle Elemente in `xs` paarweise verschieden sind.

Im folgenden sehen Sie drei Programme, die dieses Problem lösen sollen.[2] Welche Programme sind korrekt? Wenn ein Programm nicht korrekt ist, geben Sie ein Gegenbeispiel. Bewerten Sie von jedem Programm (egal ob es korrekt ist) die Effizienz, indem Sie die Anzahl von Vergleichen abschätzen, die es im schlimmsten Fall benötigt, in Abhängigkeit von der Listenlänge n.

```
-- erstes Programm
alle_verschieden_1::Ord t => [t] -> Bool
alle_verschieden_1 []  = True
alle_verschieden_1 [x] = True
alle_verschieden_1 (x1:x2:xs) | x1 /= x2  = alle_verschieden_1 (x2:xs)
                              | otherwise = False

-- zweites Programm
alle_verschieden_2::Ord t => [t] -> Bool
alle_verschieden_2 []     = True
alle_verschieden_2 (x:xs) = eindeutig && alle_verschieden_2 xs
  where eindeutig = filter (==x) xs == []

-- drittes Programm
alle_verschieden_3::Ord t => [t] -> Bool
alle_verschieden_3 = alle_verschieden_1 . sort

-- Sortieren durch Verschmelzen
sort :: Ord t => [t] -> [t]
sort [] = []
sort l@[_] = l -- Liste mit 1 Element
sort l = verschmelze (sort l1) (sort l2)
   where (l1,l2) = splitAt halb l
         halb = length l `div` 2
verschmelze :: Ord t => [t] -> [t] -> [t]
verschmelze l1 [] = l1
verschmelze [] l2 = l2
verschmelze l1@(x:xs) l2@(y:ys)  | x <= y    = x: verschmelze xs l2
                                 | otherwise = y: verschmelze l1 ys
```

95. Die Conway-Folge, Programmieraufgabe, 0 Punkte

Diese Folge entsteht, indem man in einer Zahlenfolge gleiche Zahlen zusammenfasst, in Gruppen laut vorliest und dann die ausgesprochenen Zahlen hintereinanderschreibt. Zum Beispiel würde man 33114555 als „zwei Dreien, zwei Einsen, eine Vier, drei Fünfen“ lesen und als 23 21 14 35 schreiben. Wenn man diese Operation wiederholt anwendet, erhält man immer längere Ziffernfolgen[3], zum Beispiel

$$1, 11, 21, 1211, 111221, 312211, 13112221, 1113213211, 31131211131221, \ldots$$

(a) Schreiben Sie eine Funktion `sprich:: [Int] -> [Int]` für diese Operation. Beispiel: `sprich [3,3,1,1,4,5] = [2,3,2,1,1,4,1,5]`.

(b) Schreiben Sie eine Funktion `istGültig:: [Int] -> Bool`, die feststellt, ob eine Zahlenfolge im Wertebereich der Funktion `sprich` ist.

[2] `http://www.inf.fu-berlin.de/lehre/WS16/INFA/alleverschieden.hs`
[3] `http://mathworld.wolfram.com/LookandSaySequence.html`

Informatik A, WS 2016/17 — Klausur

Abgabe bis Freitag, 17. Februar 2017, 9:45 Uhr (90 Minuten, 40 Punkte)

1. Prädikatenlogik, 10 Punkte

   Wir bezeichnen mit L_n die Menge der Listen $x = [x_0, x_1, \ldots, x_{n-1}]$ vom Typ `[Int]` mit Länge n und mit $\mathbb{N} = \{0, 1, 2, \ldots\}$ die Menge der natürlichen Zahlen.

   (a) (3 Punkte) Beschreiben Sie die Bedeutung des folgenden dreistelligen Prädikats f in Worten.

   $$f(x,y,z) :\Leftrightarrow \exists m,n \in \mathbb{N}\colon x \in L_m \ \wedge\ y \in L_n \ \wedge\ z \in L_{m+n}$$
   $$\wedge\ \forall i \in \mathbb{Z}\colon \big((0 \le i < m \to x_i = z_i) \ \wedge\ (0 \le i < n \to y_i = z_{m+i})\big)$$

   (b) (2 Punkte) Geben Sie ein Beispiel von Listen x, y, z, wo das Prädikat $f(x, y, z)$ erfüllt ist, und ein Beispiel, wo es nicht erfüllt ist,

   (c) (2 Punkte) Drücken Sie den Inhalt des Prädikats $f(x, y, z)$ in Haskell-Notation aus. Sie können dabei unendliche Listen außer Acht lassen.

   (d) (Zusatzfrage, 2 Zusatzpunkte) Wie wirkt sich die Möglichkeit unendlicher Listen auf die Antwort der vorigen Frage aus?

   (e) (3 Punkte) Welche Variablen kommen im folgenden Ausdruck frei vor?

   $$\forall i \in \mathbb{Z}\colon \big((0 \le i < m \to x_i = z_i) \ \wedge\ \exists j \in \mathbb{Z}\colon (0 \le j < n \wedge y_j = z_i)\big)$$

2. Programmieren, 10 Punkte

   (a) Listen ausdünnen, 3 Punkte

   Schreiben Sie eine Funktion `jedesZweite`, die jedes zweite Element aus einer Liste streicht, beginnend mit dem zweiten Element, und nur die Elemente an den ungeraden Stellen behält. Beispiele:

   ```
   jedesZweite [4,2,7,8,9,-1,7,2] = [4,7,9,7]
   jedesZweite "Asternplatz" = "Atrpaz"
   ```

   Überlegen Sie selbst, welchen Datentyp die Funktion hat, und schreiben Sie eine Typdeklaration.

   (b) Zinsberechnung, 3 Punkte

   Die XYZ-Bank zahlt für Anlagebeträge bis zu einem Schwellwert s von 20.000 Euro $p_1 = 3\,\%$ Zinsen pro Jahr; für darüberhinausgehende Beträge zahlt sie nur $p_2 = 1\,\%$ Zinsen. Schreiben Sie eine Funktion `zinsen:: Float -> Float -> Float -> Float -> Float`, sodass der Aufruf

   ```
   zinsen s p1 p2 x
   ```

   den Zinsbetrag für einen Anlagebetrag von `x` Euro berechnet. Für das Beispiel müsste man also `zinsen 20000 3.0 1.0 x` aufrufen. Beispiele:

   ```
   zinsen 20000 3.0 1.0  1000 =  30
   zinsen 20000 3.0 1.0 20000 = 600
   zinsen 20000 3.0 1.0 21000 = 610
   zinsen 20000 3.0 1.0 22000 = 620
   ```

   Geben Sie wie immer für alle Hilfsfunktionen Typdeklarationen an.

(c) Kontobewegungen, 4 Punkte
Schreiben Sie eine Funktion

```
endstand :: Float -> [Kontobewegung] -> Float,
```

die ausgebend von einem Ausgangskontostand und einer Liste von Kontobewegungen den Endstand berechnet. Kontobewegungen sind so definiert:

```
data Kontobewegung = Einzahlung Float
                   | Auszahlung Float
                   | Zinszahlung {s, p1, p2 :: Float}
```

Die Werte bei einer `Zinszahlung` sind die ersten drei Parameter der Funktion `zinsen` von Aufgabe (b). Bei einer `Zinszahlung` sollen die Zinsen auf den *augenblicklichen* Kontostand berechnet und gutgeschrieben werden.

3. Strukturelle Induktion, 10 Punkte

Beweisen Sie die Gleichung

```
map f (a ++ b) = map f a ++ map f b
```

für endliche Listen a und b unter Verwendung folgender Definitionen:

```
map :: (a -> b) -> [a] -> [b]
map f []     = []                    -- map.1
map f (x:xs) = f x : map f xs        -- map.2

(++) :: [a] -> [a] -> [a]
[]     ++ ys  = ys                   -- (++.1)
(x:xs) ++ ys  = x : (xs ++ ys)       -- (++.2)
```

Begründen Sie jeden Schritt, und geben Sie an, welche Definitionsgleichung (z. B. `(++.1)` oder `map.2`) oder welche anderen Eigenschaften Sie verwenden.

(a) (Zusatzfrage, 2 Zusatzpunkte) Ist die oben definierte Funktion `map` strikt im ersten Argument? Begründen Sie Ihre Antwort.

(b) (Zusatzfrage, 2 Zusatzpunkte) Ist sie strikt im zweiten Argument? Begründen Sie Ihre Antwort.

4. Entwurf eines Schaltnetzes, 10 Punkte

Gesucht ist ein Schaltnetz mit drei Eingängen $a, b, c \in \{0, 1\}$ und zwei Ausgängen $u, v \in \{0, 1\}$. Der Ausgang u soll genau dann **0** sein, wenn alle drei Eingänge den gleichen Wert haben. Der Ausgang v ist genau dann **1**, wenn die Zahl $4a + 2b + c$ mindestens 3 ist.

(a) (4 Punkte)
Beschreiben Sie die Ausgänge u und v durch logische Ausdrücke.

(b) (6 Punkte) Konstruieren ein Schaltnetz und zeichnen Sie es.
Verwenden Sie dabei die DIN-Symbole für die Gatter. Gatter mit mehr als zwei Eingängen sind nicht erlaubt.
Schreiben Sie zum Ausgang jedes Gatters (außer zu u und v) einen Booleschen Ausdruck in den Eingabevariablen, der den Wert charakterisiert.

Informatik A, WS 2016/17 — Nachklausur

Abgabe bis Dienstag, 28. März 2017, 11:45 Uhr (90 Minuten, 40 Punkte)

---

1. Aussagenlogik, 10 Punkte

   (a) Umformung, 6 Punkte

   Beweisen Sie folgende Äquivalenzen, entweder durch Anwendung von Umformungsregeln oder durch Vergleichen der Wahrheitstafeln:

   $$(p \wedge s) \Rightarrow q \;\equiv\; (p \Rightarrow q) \vee (s \Rightarrow q) \tag{1}$$
   $$x \Leftrightarrow y \;\equiv\; (\neg x) \Leftrightarrow (\neg y) \tag{2}$$

   (b) NOR, 4 Punkte

   Der Junktor $\bar{\vee}$ (weder-noch, engl. „nor“) ist definiert als

   $$a \mathbin{\bar{\vee}} b \;\equiv\; \neg(a \vee b).$$

   Ist der Junktor $\bar{\vee}$ kommutativ?

   Ist er assoziativ?

   Geben Sie jeweils einen Beweis oder ein Gegenbeispiel.

2. Induktion, 10 Punkte

   (a) (4 Punkte) Bestimmen Sie alle Paare (k, l) von ganzen Zahlen, für die die folgende Gleichung gültig ist:

   ```
   (drop k) . (drop l) = drop (k+l)
   ```

   (Die Möglichkeit des Überlaufs bei der Berechnung von `k+l` können Sie außer Acht lassen.) *Geben Sie eine Begründung* für die Fälle, wo die Gleichung nicht gilt.

   (b) (6 Punkte) Für die Fälle, wo die Gleichung gilt, beweisen Sie sie unter Verwendung folgender Definitionen:

   ```
   drop :: Int -> [a] -> [a]
   drop n xs  | n <= 0 =  xs                  -- drop.0
   drop _ []           =  []                  -- drop.1
   drop n (x:xs)       =  drop (n-1) xs       -- drop.2

   (.) :: (b -> c) -> (a -> b) -> (a -> c)
   (f . g) x = f (g x)                        -- punkt.0
   ```

   Begründen Sie jeden Schritt, indem Sie angeben, welche Definitionsgleichung (drop.0 oder drop.1 oder drop.2 oder punkt.0) oder welche anderen Eigenschaften sie verwenden.

3. Programmieren, 10 Punkte

   (a) Listen verzahnen, 3 Punkte
   Schreiben Sie eine Funktion `jedesZweite`, die zwei Listen im Reißverschlusssystem verzahnt. Die Elemente der ersten Liste kommen an die ungeraden Positionen des Ergebnisses, und die Elemente der zweiten Liste füllen die geraden Positionen. Die Liste endet, sobald eine der Eingabelisten erschöpft ist.
   Beispiele:

   ```
   jedesZweite [4,7,9,7] [2,8,-1,2,3] = [4,2,7,8,9,-1,7,2]
   jedesZweite "Atrpazifik" "senlt" = "Asternplatz"
   ```

   Überlegen Sie selbst, welchen Datentyp die Funktion hat, und *schreiben Sie eine Typdeklaration.*

   (b) Wie viele gleiche? 7 Punkte
   Schreiben Sie eine Funktion

   `wieVieleGleiche:: Eq a => [a] -> Int`,

   die bestimmt, wie oft das häufigste Element in einer Liste vorkommt.
   Beispiele:

   `wieVieleGleiche [9,4,2,4] = 2`, weil das Element 4 zweimal vorkommt.
   `wieVieleGleiche [4,9,2,14] = 1`, weil jedes Element nur einmal vorkommt.

   Beachten Sie, dass die Elemente der Eingabeliste nicht zur Typklasse `Ord`, sondern nur zur Typklasse `Eq` gehören!
   Sie können beliebige Funktionen aus den Standardbibliotheken zu Hilfe nehmen. (Bei der Bewertung wird nicht darauf geachtet, ob sie die richtigen Modulnamen für den Import der Bibliotheksfunktionen schreiben.)

   (c) Zusatzfrage (2 Zusatzpunkte). Analysieren Sie die asymptotische Laufzeit Ihres Programms zu Aufgabe (b).

4. Entwurf eines Schaltnetzes, 10 Punkte

   Gesucht ist ein Schaltnetz mit drei Eingängen $a, b, c \in \{0, 1\}$ und zwei Ausgängen $u, v \in \{0, 1\}$.

   Der Ausgang u soll gleich a sein, falls $b = 0$ ist; andernfalls soll er gleich c sein.

   Wenn alle drei Eingänge den gleichen Wert haben, soll der Ausgang v gleich 0 sein; andernfalls soll v gleich b sein.

   (a) (4 Punkte) Beschreiben Sie die Ausgänge u und v durch logische Ausdrücke (nur mit Junktoren, nicht mit if-then-else-Ausdrücken).

   (b) (6 Punkte) Konstruieren ein Schaltnetz und zeichnen Sie es.
   Verwenden Sie dabei die DIN-Symbole für die Gatter. Gatter mit mehr als zwei Eingängen sind nicht erlaubt.
   Schreiben Sie zum Ausgang jedes Gatters (außer zu u und v) *einen Booleschen Ausdruck* in den Eingabevariablen, der den Wert charakterisiert.