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.

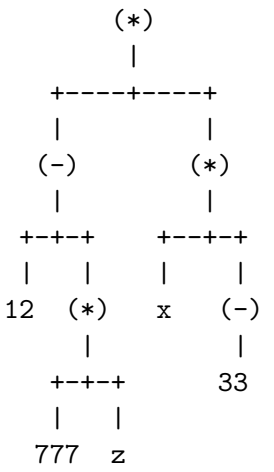
    Formulieren Sie die letzte Aussage in Worten.

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`