Praktische Informatik I | WiSe 98/99 | Prof. Bernd Krieg-Brückner

Shi Hui
Kolyang
Axel Lankenau
Christoph Lüth
Lutz Zegartowski

4. Übungsblatt

Ausgabe: 03.12.98
Abgabe: 11.01. - 15.01.99

---

1 *Das kommt davon, wenn man nicht artig ist* — *Wichtung: 30%*

Der kleine BKB hat sich zum Nikolaus einen funktionalen Taschenrechner gewünscht. Da der Nikolaus wußte, daß BKB dieses Jahr nicht so artig war, hat er sich gedacht, daß er dem Kleinen lieber ein Haskell-Buch mitbringt und daß der Bub seinen Rechner gefälligst selber implementieren soll. Ihr sollt Klein-BKB dabei helfen, seinen Wunsch wahr werden zu lassen: Definiert einen Datentyp `IntExp` für einfache arithmetische Ausdrücke, dessen Werte entweder ganze Zahlen (repräsentiert als `I` *n*, mit $n \in Z$) oder zusammengesetzte Ausdrücke der Form `Z IntExp Op IntExp` sein können. Dabei ist `Op` ein ebenfalls zu definierender Datentyp, der die Werte `Plus`, `Minus`, `Mal` und `Durch` annehmen kann.

1. Definiere die geforderten Datentypen. Ergänze sie nach Belieben um weitere Werte.

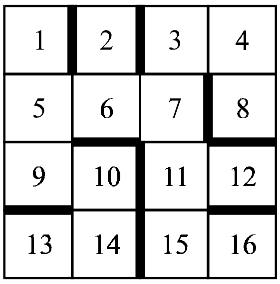
2. Implementiere eine Funktion
   `calc :: IntExp -> IntExp`,
   die einen arithmetischen Ausdruck vom Typ `IntExp` soweit wie möglich auswertet. Das Resultat soll die Form `I` *n* besitzen. So wird `calc (Z (I 6) Mal (I 9))` zum Beispiel zu `54` ausgewertet.

3. Während das Ergebnis recht übersichtlich daherkommt, ist der ursprüngliche Ausdruck doch einigermaßen unleserlich. Schreibe eine Funktion
   `pretty :: IntExp -> String`,
   die einen arithmetischen Ausdruck vom Typ `IntExp` schön als Text darstellt. So sollte beispielsweise `pretty (Z (I 6) Mal (I 9))` zu `"(6) * (9)"` werden.

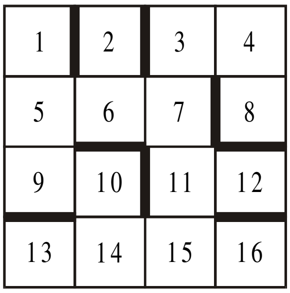
2 *Santa Hugs und Rudolph, das Rentier...* — *Wichtung: 40%*

Auch das noch! Kurz vor Weihnachten hat sich Rudolph, das Rentier, im Wald verlaufen. Santa Claus ist ratlos; wenn er doch nur wüßte, wie er zu Rudolph finden kann; schließlich muß der Schlitten vollständig angespannt werden... Da hat er eine Idee: Sein Halbbruder, *Santa Hugs*, könne ihm bestimmt helfen, den kürzesten Weg zu Rudolph durch den Wald zu finden. Dieser erklärt sich auch prompt bereit und macht folgenden Vorschlag:

- Man zerlege den Wald in einzelne Planquadrate.
- Jedes Quadrat erhält eine eindeutige Nummer.
- Man definiere die begehbaren Wege als Übergänge zwischen den einzelnen Quadraten.



Wald *a*



Wald *b*

1. Da *Santa Hugs* ein Verehrer von Listen ist, macht er für Wald *a* den Vorschlag, eine Haskell-Funktion `bringTogether :: Int -> Int -> [Int]` zu implementieren, die einen beliebigen Startpunkt von Santa Claus und eine beliebige Position von Rudolph übergeben bekommt und eine Wegebeschreibung zurückliefert. Dabei soll eine Funktion `next :: Int -> [Int]` benutzt werden, die das Labyrinth des Waldes repräsentiert (z.B. `next 6 = [2,5,7]`). Für welche Fälle arbeitet der Algorithmus korrekt?
2. Verändere Deinen Algorithmus so, daß er nun auch für Wald *b* funktioniert und den kürzesten Weg zwischen zwei Punkten liefert.

3 *Santa Hugs vs. BKB* — *Wichtung: 30%*

Dank der Hilfe von Santa Hugs konnten Rudolph und der Weihnachtsmann wieder zusammengeführt werden. Allerdings hat der kleine BKB einige Schwierigkeiten mit der Listen-Lösung. Sein Argument *„Du sieht die Bäume vor lauter Listen nicht“* hat Santa Hugs nachdenklich gemacht und er überlegt nun, wie man das Problem aus Aufgabe 2 mittels *n-ärer Bäume* lösen kann. Ihr sollt ihm dabei helfen...

1. Definiert einen rekursiven Datentyp `Baumwald`, der den in Aufgabe 2 dargestellten Wald *a* als Baumstruktur darstellt.
2. Die Wurzel des Baumes soll das Feld 1 sein.
3. Verwendet eine Funktion `child`, die zu jedem Blatt die „Nachfolge-Blätter“ zurückliefert.
4. Die Blätter des Baumes sind die jeweiligen Felder, die vom aktuellen Blatt aus erreicht werden können.
5. Schreibt eine Suchfunktion, die einen Weg von Blatt zu Blatt findet, damit Santa Claus und Rudolph wieder zueinander finden. (Es ist ebenfalls hilfreich, eine ähnliche Strategie wie in Aufgabe 2 zu entwickeln und zu verwenden.)
6. Diskutiert die Vor- und Nachteile der Listen- und der Baummethode. Gibt es Gemeinsamkeiten bzw. Unterschiede? Welche?