Algorithmen und Programmieren III, WS 1999/2000 — 10. Übungsblatt

Abgabe bis Dienstag, 1. Feber 2000.

49. (12 Punkte) Betrachten Sie die folgende Haskell-Funktion:

```
istTeil (x:xs) (y:ys) = ( (x==y) && istTeil xs ys) ) || istTeil (x:xs) ys
istTeil [] _ = True
istTeil (_:_) [] = False
```

   (a) (5 Punkte) Geben Sie (formal oder umgangssprachlich, aber präzise) an, was die Funktion `istTeil` berechnet.

   (b) (0 Punkte) Bestimmen Sie experimentell oder durch Analyse die Laufzeit der Funktion in Abhängigkeit von der Länge der Argumente. Für welche Argumente dauert die Berechnung der Funktion am längsten?

   (c) (0 Punkte) Untersuchen Sie die vorige Frage auch für die Variante, wo die beiden Teile der Disjunktion (`||`) in der ersten Zeile vertauscht sind.

   (d) (3 Punkte) Beweisen Sie die folgenden logischen Implikationen:

$$\begin{aligned} \texttt{istTeil xs ys} &\implies \texttt{istTeil xs (y:ys)} \\ \texttt{istTeil (x:xs) ys} &\implies \texttt{istTeil xs ys} \end{aligned}$$

   (Für die zweite Aussage können Sie unter anderem Induktion nach der Länge der Liste im zweiten Argument verwenden.)

   (e) (4 Punkte) Die folgende Variante der ersten Zeile führt auf eine äquivalente Funktion:

```
istTeil (x:xs) (y:ys)
    | x==y  = istTeil xs ys
    | x/=y  = istTeil (x:xs) ys
```

   Erklären Sie diesen Sachverhalt (in Worten), und beweisen Sie ihn formal. (Die Aussagen aus der vorigen Aufgabe könnten dabei hilfreich sein.)

   (f) (0 Punkte) Untersuchen Sie diese letzte Variante auf ihre Effizienz.

50. (7 Punkte) Der Computer SUMO-1 ist eine verbesserte Variante von SUMO-0 aus Aufgabe 39. Die wichtigste Neuerung ist, dass eine Variable auch *mehrmals* auf der linken Seite einer Zuweisung auftreten kann. Weiters gibt es eine Druck-Anweisung Px, die den Inhalt der Variablen Vx ausdruckt und das Programm beendet.

   Die Programmierung in SUMATRA soll durch einen Plausibilitätstest unterstützt werden. Dazu sollen Anweisungen, die das gedruckte Endergebnis nicht beeinflussen können, identifiziert werden. Schreiben Sie ein Programm für diese Aufgabe.

51. (5 Punkte) Erweitern Sie die Spezifikation für den abstrakten Datentype *Menge* (von ganzen Zahlen) aus der Vorlesung mit den Operationen Einfügen, Entfernen, Elementtest, und Erstellen einer leeren Menge um das Erstellen eines *Iterators* mit den Operationen *next* und *hasNext* (wie im Java-Interface `Iterator`). Spezifizieren Sie diese Operationen formal.

52. (0 Punkte) Leiten Sie aus der algebraischen Spezifikation für Mengen

$$\begin{aligned} \mathit{istenthalten}(x, \mathit{leer}) &= \mathit{falsch} \\ \mathit{istenthalten}(x, \mathit{einfüge}(x, M)) &= \mathit{wahr} \\ \mathit{istenthalten}(x, \mathit{einfüge}(y, M)) &= \mathit{istenthalten}(x, M), \qquad \text{für } x \neq y \\ \mathit{istenthalten}(x, \mathit{lösche}(x, M)) &= \mathit{falsch} \\ \mathit{istenthalten}(x, \mathit{lösche}(y, M)) &= \mathit{istenthalten}(x, M), \qquad \text{für } x \neq y \end{aligned}$$

   (mit den Signaturen von *einfüge*, *lösche*, *leer*, und *istenthalten* wie in der Vorlesung) durch Umformungen folgende Identität her: Für $x \neq y$ gilt

$$\mathit{istenthalten}(u, \mathit{lösche}(x, \mathit{einfüge}(y, M))) = \mathit{istenthalten}(u, \mathit{einfüge}(y, \mathit{lösche}(x, M))).$$

53. (0 Punkte) Erweitern Sie die algebraische Spezifikation von Mengenoperationen aus der vorigen Aufgabe um die Funktionen *Vereinigung* und *Durchschnitt* (von je zwei Mengen).