5. Übungsblatt zu Programmierung 1, WS 2010/11

Prof. Dr. Gert Smolka, Christian Doczkal, M.Sc.
www.ps.uni-sb.de/courses/prog-ws10/

Lesen Sie im Buch: Kapitel 5

Aufgabe 5.2 Schreiben Sie eine Prozedur *sorted* : *int list* → *bool*, die testet, ob eine Liste aufsteigend sortiert ist. Verwenden Sie dabei keine Hilfsprozedur.

Aufgabe 5.3 Schreiben Sie eine Prozedur *perm* : *int list* → *int list* → *bool*, die testet, ob zwei Listen bis auf die Anordnung ihrer Elemente gleich sind. Verwenden Sie dabei die Prozedur *isort* und die Tatsache, dass *int list* ein Typ mit Gleichheit ist.

Aufgabe 5.4 Schreiben Sie eine Prozedur *issort* : *int list* → *int list*, die eine Liste sortiert und dabei Mehrfachauftreten von Elementen eliminiert. Beispielsweise soll für $[3,1,3,1,0]$ die Liste $[0,1,3]$ geliefert werden.

Aufgabe 5.7 Deklarieren Sie eine Prozedur

$$\mathit{intPairCompare} : (\mathit{int} * \mathit{int}) * (\mathit{int} * \mathit{int}) \to \mathit{order}$$

die die lexikalische Ordnung für Paare des Typs $\mathit{int} * \mathit{int}$ darstellt. Zum Beispiel soll $[(3,4),\ (3,5),\ (4,0)]$ gemäß dieser Ordnung sortiert sein.

Aufgabe 5.8 Deklarieren Sie eine Prozedur

$$\mathit{lex}\colon\ (\alpha * \alpha \to \mathit{order})\ \to\ (\beta * \beta \to \mathit{order})\ \to\ (\alpha * \beta) * (\alpha * \beta) \to \mathit{order}$$

die zu Ordnungen für die Typen α und β die lexikalische Ordnung für den Produkttyp $\alpha * \beta$ liefert. Deklarieren Sie mit Ihrer Prozedur *lex* eine Prozedur, die die lexikalische Ordnung für Paare des Typs $\mathit{int} * \mathit{real}$ darstellt, die in der ersten Position absteigend und in der zweiten Position aufsteigend sortiert. Zum Beispiel soll $[(3,4.0),\ (2,2.0),\ (2,3.0)]$ gemäß dieser Ordnung sortiert sein.

Aufgabe 5.9 Schreiben Sie eine polymorphe Sortierprozedur, die durch Mischen sortiert.

Aufgabe 5.14 (Striktes Sortieren durch Mischen)

a) Schreiben Sie eine polymorphe Prozedur *smerge*, die zwei strikt sortierte Listen zu einer strikt sortierten Liste kombiniert. Beispielsweise soll gelten: *smerge Int.compare* $([1,3],\ [2,3]) = [1,2,3]$.

b) Schreiben Sie eine polymorphe Prozedur *ssort*, die Listen strikt sortiert. Beispielsweise soll *ssort Int.compare* $[5,3,2,5] = [2,3,5]$ gelten.

c) Machen Sie sich klar, dass es sich bei *ssort Int.compare* um eine Prozedur handelt, die zu einer Liste *xs* die eindeutig bestimmte strikt sortierte Liste liefert, die dieselbe Menge wie *xs* darstellt.

Aufgabe 5.15 Seien endliche Mengen ganzer Zahlen gemäß *Int.compare* durch strikt sortierte Listen dargestellt. Deklarieren Sie Prozeduren wie folgt:

a) *member* : *int* → *int list* → *bool* testet, ob eine Zahl Element einer Menge ist.
b) *union* : *int list* → *int list* → *int list* liefert die Vereinigung zweier Mengen.
c) *intersection* : *int list* → *int list* → *int list* liefert den Schnitt zweier Mengen.
d) *difference* : *int list* → *int list* → *int list* liefert die Differenz zweier Mengen.
e) *eqset* : *int list* → *int list* → *bool* testet, ob zwei Mengen gleich sind.
f) *subset* : *int list* → *int list* → *bool* testet für zwei Mengen X und Y, ob $X \subseteq Y$.
   Realisieren Sie *subset*
   (i) mit *member*
   (ii) mit *eqset* und *intersection* gemäß der Äquivalenz $X \subseteq Y \iff X = X \cap Y$.
   (iii) ohne Hilfsprozedur

Aufgabe 5.16 Seien endliche Mengen durch Listen dargestellt. Beispielsweise kann die Menge $\{1, 2, 3\}$ durch die Liste $[2, 1, 2, 3]$ dargestellt werden. Deklarieren Sie Prozeduren wie folgt:

a) *member* : *''a* → *''a list* → *bool* testet, ob ein Wert Element einer durch eine Liste dargestellten Menge ist.
b) *subset* : *''a list* → *''a list* → *bool* testet für zwei Listen *xs* und *ys*, ob die durch *xs* dargestellte Menge eine Teilmenge der durch *ys* dargestellten Menge ist.
c) *eqset* : *''a list* → *''a list* → *bool* testet, ob zwei Listen dieselbe Menge darstellen.

Aufgabe 5.17 (Potenzmenge) Seien endliche Mengen durch Listen ohne Mehrfachauftreten dargestellt. Deklarieren Sie eine Prozedur *power* : *α list* → *α list list*, die zu einer Menge X die Menge liefert, die genau die Teilmengen von X enthält. Dabei ist die Reihenfolge beliebig, aber es soll keine Mehrfachauftreten geben. Beispielsweise wäre es korrekt, wenn *power* für $[1, 2]$ die Liste $[[1, 2], [2], [1], []]$ liefert.

Aufgabe 5.19 (Liste der ersten n Primzahlen) Deklarieren Sie eine Prozedur *primelist* : *int* → *int list*, die zu $n \geq 0$ die Liste der ersten n Primzahlen in aufsteigender Ordnung liefert. Verwenden Sie die Prozedur *nextprime* aus § 3.11. Hinweis: Verwenden Sie ein Akku und berechnen Sie die Liste der ersten n Primzahlen ab einer gegebenen Primzahl.