Technische Universität München
Fakultät für Informatik
Prof. Dr. A. Knoll

SS 2006
Übungsblatt 4
2. Juni 2006

Übungen zu Einführung in die Informatik I

Aufgabe 13 Listen in OCaml

In dieser Aufgabe werden wir die Datenstruktur der Liste selbst definieren.

a) Definieren Sie rekursiv einen neuen Typ `myList` der entweder leer (`Empty`) sein kann oder ein Element (`Elem`) das aus einem Wert `'a` und einer Liste besteht.

b) Definieren Sie eine Funktion `prepend`, welche an eine vorhandene Liste einen Wert vorne anfügt. Testen Sie ihre neue Datenstruktur indem Sie eine Liste mit den Zeichen `a`, `h`, `a` aufbauen.

c) Schreiben Sie eine (rekursive) Funktion `append`, welche an eine vorhandene Liste einen Wert hinten anhängt. Testen Sie Ihre Funktion indem Sie das Zeichen `!` an die Liste der letzten Teilaufgabe anhängen.

d) Schreiben Sie eine (rekursive) Funktion `isElement`, welche für eine Liste und einen Wert testet, ob der Wert in der Liste enthalten ist. Testen Sie Ihre neudefinierte Funktion für die Liste aus den vorherigen Teilaufgaben und die Zeichen `x`, `h` und `a`.

e) Schreiben Sie eine (rekursive) Funktion `insert`, welche einen Wert in einer Liste **sortiert** einfügt. Die kleinsten Elemente sollen dabei am Ende der Liste stehen. Testen Sie Ihre Funktion mit den Werten 23, 12, 42, 24, 1, 23.

Aufgabe 14 Das Prinzip der Induktion

In dieser Aufgabe lernen Sie die Induktion kennen, ein mathematisches Hilfsmittel, das im weiteren Verlauf der Übungen noch häufig verwendet werden wird.

Es sei für $x \in \mathbb{N}$ die boolesche Funktion `gerade` (x), die ermittelt, ob x durch 2 teilbar ist oder nicht, wie folgt spezifiziert:

$$\texttt{gerade}\ (x) = (\mathrm{mod}(x,2) \stackrel{?}{=} 0)$$

(Hinweis: Die Funktion $\mathrm{mod}(x,2)$ berechnet den Rest bei ganzzahliger Division von x durch 2.) Darüber hinaus sei die Funktion `gerade`$'$ (x) durch

$$\texttt{gerade}'\ (x) = \begin{cases} \texttt{True}, & \text{falls } x = 0 \\ \neg\ (\texttt{gerade}'\ (\texttt{x} - 1)), & \text{falls } x \in \mathbb{N} \setminus \{0\} \end{cases}$$

gegeben.

Aufgabenstellung: Zeigen Sie mit Hilfe der Induktion, dass die beiden Funktionen `gerade` (x) und `gerade`$'$ (x) äquivalent sind.

Aufgabe 15 Induktion über den natürlichen Zahlen

Seien $n \in \mathbb{N}_0, k \in \mathbb{N}$ und das Prädikat

$$divides(k,n) = (\exists m \in \mathbb{N}_0 : m \cdot k = n)$$

gegeben. Die Funktion

$$divides_{Imp}(k,n) = \begin{cases} true & n = 0 \\ false & 0 < n < k \\ divides_{Imp}(k, n-k) & n \geq k \end{cases}$$

sei eine Implementierung des Prädikates *divides*.

a) Zeigen Sie durch vollständige Induktion über n, dass

$$divides(k,n) = divides_{Imp}(k,n)$$

für $n \in \mathbb{N}_0, k \in \mathbb{N}$ gilt. (**Hinweis:** Beachten Sie, dass hier mehrere Induktionsanfänge zu bearbeiten sind!)

b) Definieren Sie eine rekursive Funktion in OCaml-Syntax, die die Anzahl der rekursiven Aufrufe der Funktion $divides_{Imp}$ berechnet.

Aufgabe 16 (H) Induktion: `take` und `drop` (4+6 Punkte)

Seien `take m ls` bzw. `drop n ls` die beiden Funktionen, die aus einer Liste `ls` die ersten `m` Elemente von `ls`, bzw. die Liste ohne die ersten `n` Elemente, zurückgeben. Es gelte ausserdem: `take 0 ls = [], take n [] = [], drop 0 ls = ls, drop n [] = []`.

a) Implementieren Sie `take` und `drop`.

b) Beweisen Sie mittels vollständiger Induktion, dass $\forall$ `n` $\in \mathbb{N}_0$, `ls` $\in [a]$ gilt:

$$\texttt{concat}(\texttt{take n ls}, \texttt{ drop n ls}) = \texttt{ls}$$

Hinweis: Führen Sie den Beweis über `n` und nicht wie intuitiv anzunehmen über `ls`.