TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

WS 10/11

Einführung in die Informatik II **Übungsblatt 14**

Univ.-Prof. Dr. Andrey Rybalchenko, A. Herz, K. Apinis 11.2.2011

Dieses Blatt behandelt Wiederholungsaufgaben für die Klausur.

Aufgabe (Strukturelle Induktion)

Gegeben sei die folgende Prozedur:
$p : \mathcal{L}(X) \to \mathcal{L}(X)$
$p\ nil = nil$
$p\ (x :: nil) = x :: nil$
$p\ (x :: y :: xr) = x :: (p\ xr)$

a) Geben Sie die Terminierungsfunktion an.

b) Beweisen sie mittels struktureller Induktion: $\forall xs \in \mathcal{L}(X).|pxs| = \lceil \frac{|xs|}{2} \rceil$

Aufgabe (SML Strukturen)

```
signature Complex = sig
  type complex (*two reals*)
  val add: complex → complex → complex
  val sub: complex → complex → complex
  val conjugate: complex → complex
  val multiply: complex → complex → complex
  val abs: complex → real
end
```

Aufgabe 15.4 (Generatoren)

Ein Generator für eine Folge x1, x2, ... von Werten eines Typs t ist eine Prozedur `unit → t`, die beim n-ten Aufruf das Folgenglied xn liefert.

a) Schreiben Sie einen Generator square für die Folge 1, 4, 9, ... der Quadratzahlen.

b) Schreiben Sie eine Prozedur `newSquare : unit → unit → int`, die bei jedem Aufruf einen neuen Generator für die Folge der Quadratzahlen liefert.

c) Schreiben Sie eine Prozedur `newGenerator : (int → 'a) → unit → 'a`, die zu einer Prozedur f einen Generator für die Folge f 1, f 2, f 3, ... liefert.

d) Schreiben Sie mithilfe der Prozedur newGenerator einen Generator cube für die Folge 1, 8, 27, ... der Kubikzahlen.

e) Schreiben Sie mithilfe der Prozedur newGenerator eine Prozedur newCube, die bei jedem Aufruf einen neuen Generator für die Folge der Kubikzahlen liefert.

Aufgabe (Strukturelle Induktion 2)

Gegeben seien folgende Prozeduren:
$even : int \to bool$
$even\ 0 = true$

$even\ n = odd(n-1)$

$odd : int \to bool$
$odd\ 0 = false$
$odd\ n = even(n-1)$

a) Beweisen sie mittels struktureller Induktion: $\forall n \in \mathbb{N} : even\ n = (n\ mod\ 2 = 0)$

Aufgabe Ausnahmen

Gegeben sei die Datenstruktur für einen (unsortierten) Binärbaum:

```
datatype tree = Leaf | Node of tree * int * tree
```

a) Schreiben sie eine Prozedur `findSubtree n  tr : int → tree → tree`, die den Unterbaum zurückgibt ab dem Knoten der die Zahl n enthält. Falls die Zahl nicht im Baum vorkommt soll eine Ausnahme geworfen werden `NotFound: int` welche die gesuchte Zahl enthält.

   Da der Baum nicht sortiert ist und deshalb die selbe Zahl in mehrere Knoten vorkommen kann soll die Funktion für diesen Fall eine Ausnahme werfen `MultipleFindings: tree list` die die Liste aller Teilbäume in denen die gesuchte Zahl vorkommt enthält.

b) Schreiben sie eine Prozedur `findSubtree_v2 : int → tree → tree`, die nach einer Zahl im Baum sucht und einen leeren Baum zurückgibt falls die Zahl im Baum nicht gefunden wird bzw. den ersten gefundenen Unterbaum falls mehrere treffer existieren.