TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

WS 11/12

Einführung in die Informatik II **Übungsblatt 7**

Univ.-Prof. Dr. Andrey Rybalchenko, M.Sc. Ruslán Ledesma Garza 13.12.2010

Dieses Blatt behandelt Kapitel 6.1 – 6.6, 15.1 – 15.3, und 15.8 aus dem Buch zur Vorlesung. Lesen Sie diese Kapitel!

Aufgabe 6.2 Deklarieren Sie eine Prozedur `scale : real → shape → shape`, die ein Objekt gemäß einem Faktor skaliert (d.h. vergrößert oder verkleinert). Beispielsweise soll `scale 0.5 (Square 3.0) = Square 1.5` gelten.

Aufgabe 6.3 In §4.6.3 haben Sie gelernt, dass das Konditional eine abgeleitete Form ist. Wissen Sie noch, auf welchen Ausdruck der Kernsprache ein Konditional **if** `e1` **then** `e2` **else** `e3` reduziert?

Aufgabe 6.5 Deklarieren Sie eine Prozedur `vars : exp → var list`, die zu einem Ausdruck eine Liste liefert, die die in dem Ausdruck vorkommenden Variablen enthält. Orientieren Sie sich an der Prozedur `subexps`.

Aufgabe 6.7

Deklarieren Sie eine Prozedur `check : exp → exp → bool`, die für zwei Ausdrücke e und e' testet, ob e ein Teilausdruck von e' ist.

Aufgabe 6.8 Schreiben Sie eine Prozedur `instantiate : env → exp → exp`, die zu einer Umgebung V und einem Ausdruck e den Ausdruck liefert, den man aus e erhält, indem man die in e vorkommenden Variablen gemäß V durch Konstanten ersetzt. Beispielsweise soll für die oben deklarierte Umgebung env und den Ausdruck `A(V "x", V "y")` der Ausdruck `A(C 5, C 3)` geliefert werden. Orientieren Sie sich an der Prozedur eval.

Aufgabe 6.10* (Konstruktordarstellung natürlicher Zahlen) In dieser Aufgabe stellen wir die natürlichen Zahlen mit den Werten des Konstruktortyps

datatype `nat = O | S` **of** `nat`

dar: $0 \mapsto$ `O`, $1 \mapsto$ `S O`, $2 \mapsto$ `S (S O)`, $3 \mapsto$ `S (S (S O))`, und so weiter.

a) Deklarieren Sie eine Prozedur `code : int → nat`, die die Darstellung einer natürlichen Zahl liefert.

b) Deklarieren Sie eine Prozedur `decode : nat → int`, sodass `decode (code n) = n` für alle $n \in \mathbb{N}$ gilt.

c) Deklarieren Sie für `nat` kaskadierte Prozeduren `add`, `mul`, `sub` und `less`, die den Operationen $+$, $*$, $-$ und $<$ für natürliche Zahlen entsprechen. Verwenden Sie dabei keine Operationen für `int`. Werfen Sie eine Ausnahme, falls es kein sinnvolles Ergebnis gibt.

Aufgabe 6.12

Schreiben Sie eine Prozedur `test : int → bool`, die testet, ob das Quadrat einer ganzen Zahl im darstellbaren Zahlenbereich liegt.

Aufgabe 6.13

Führen Sie zweistellige Sequenzialisierungen (e1; e2) auf Abstraktionen und Applikationen zurück.

Aufgabe 6.14

```
exception Double

fun mask compare p =
  case compare p of
      EQUAL ⇒ raise Double
    | v     ⇒ v

fun testDouble compare xs =
  ( List.sort (mask compare) xs;
    false
  ) handle Double ⇒ true
```

Schreiben Sie die Prozedur `testDouble` so um, dass die Ausnahme `Double` und die Hilfsprozedur `mask` mithilfe eines Let-Ausdrucks lokal deklariert werden.

Aufgabe 6.16 Schreiben Sie eine Prozedur `append : 'a mylist → 'a mylist → 'a mylist` die zwei gemäß des Typkonstruktors mylist (**datatype** `'a mylist = Nil | Cons` **of** `'a * 'a mylist`) dargestellte Listen konkateniert.

Aufgabe 15.6

Vervollständigen Sie die Deklaration **val** `(count, inc, reset) =` so, dass sie einen eingekapselten Zähler mit dem Anfangswert 0 und drei Prozeduren wie folgt liefert:

- `count : unit → int` liefert den Wert des Zählers.
- `inc : unit → unit` erhöht den Wert des Zählers um 1.
- `reset : unit → unit` setzt den Zähler auf 0 zurück.

Aufgabe 15.21

Schreiben Sie eine Prozedur `length : 'a list → int`, die mit einer Schleife die Länge einer Liste bestimmt. Verwenden Sie die vordeklarieren Prozeduren `null`, `tl` und `not`.