## 14. Übungsblatt zu Programmierung

Prof. Gert Smolka, Thorsten Brunklaus
www.ps.uni-sb.de/courses/prog-ws00/

Abgabe: 23. Februar 2001 vor der Vorlesung (per Email)

**Allgemeine Hinweise:** Die Übungsblätter sollen in Zweiergruppen bearbeitet werden. Die Lösungen der Aufgaben schicken Sie bitte bis Freitag vor der Vorlesung per Email an Ihren Übungsgruppenleiter. Jede Gruppe soll nur eine Lösung einreichen, versehen mit den Namen und den Matrikelnummern der Gruppenmitglieder.

**Aufgabe 14.1: Größe von imperativen Listen (8+2)** Schreiben Sie eine Prozedur

```
size : 'a ilist -> int
```

die die Größe einer imperativen Liste liefert (das ist die Anzahl der Referenzen der Liste). Schreiben Sie dazu zuerst eine Prozedur

```
reflist : 'a ilist -> 'a ilist ref list
```

die die Liste aller Referenzen einer imperativen Liste liefert. Beide Prozeduren sollen auch für zyklische Listen funktionieren. Geben Sie die Laufzeit Ihrer Prozedur `size` an.

**Aufgabe 14.2: Größter gemeinsamer Teiler (2+8)** Die Prozedur

```
fun gcd(x,y) = if x<=y
               then if y<=x then x else gcd(x, y-x)
               else gcd(x-y, y)
```

berechnet den größten gemeinsamen Teiler zweier positiven ganzen Zahlen.

(a) Schreiben Sie mithilfe einer Schleife eine nichtrekursive Variante der Prozedur.

(b) Übersetzen Sie die Prozedur in eine Liste *gcd* von V0-Befehlen, so dass das V0-Programm

```
[con x, con y] @ gcd @ [halt]
```

den größten gemeinsamen Teiler zweier positiven ganzen Zahlen $x$ und $y$ berechnet.

**Aufgabe 14.3: Übersetzung und Rückübersetzung (3+7)** Seien die folgenden Ausdrücke gegeben:

```
datatype exp =
    Con    of int             (* constant          *)
  | Add    of exp * exp       (* addition          *)
  | Sub    of exp * exp       (* subtraction       *)
  | Mul    of exp * exp       (* multiplication    *)
```

(a) Schreiben Sie eine Prozedur

```
compile : exp -> code
```

die einen Ausdruck in ein V0-Programm übersetzt, dessen Ausführung den Wert des Ausdrucks liefert.

(b) Schreiben Sie eine Prozedur

```
decompile : code -> exp
```

sodass für alle Ausdrücke *e* gilt:

```
decompile(compile e) = e
```

**Aufgabe 14.4: Fibonacci-Prozedur in V (5+5)** Übersetzen Sie die Prozedur

```
fun fib n = if n<2 then n else fib(n-2) + fib(n-1)
```

in eine Befehlssequenz für V. Nehmen Sie dabei an, dass die Befehlssequenz mit der Adresse 0 beginnend im Programmspeicher abgelegt wird.
Übersetzen Sie die Prozeduren

```
fun fibi'(n,a,b) = if n<=0 then a else fibi'(n-1, b, a+b)

fun fibi n = fibi'(n,0,1)
```

in eine Befehlssequenz für V. Nehmen Sie dabei an, dass die Befehlssequenz mit der Adresse 0 beginnend im Programmspeicher abgelegt wird. Übersetzen Sie alle Endaufrufe mit `callR`.

**Aufgabe 14.5: Verschränkte Rekursion in V (10)** Übersetzen Sie die Prozeduren

```
fun even x = if x=0 then true else odd(x-1)
and odd  x = if x=0 then false else even(x-1)
```

in eine Befehlssequenz für V. Nehmen Sie dabei an, dass die Befehlssequenz mit der Adresse 0 beginnend im Programmspeicher abgelegt wird. Übersetzen Sie alle Endaufrufe mit `callR`.