

Programmierung 1 (Wintersemester 2012/13)

Aufgaben aus den Übungsgruppen 1

(Kapitel 1)

Hinweis: Dieses Übungsblatt enthält von den Tutoren für die Übungsgruppe erstellte Aufgaben. *Die Aufgaben und die damit abgedeckten Themenbereiche sind für die Klausur weder relevant, noch irrelevant.*

1 Rekursive Prozeduren verstehen

Aufgabe 1.1

Welche mathematischen Funktionen berechnen die folgenden endrekursiven Prozeduren?

Beispiel: Die Prozedur

```
fun f (x:int) :int = x * x
```

berechnet die Funktion $f: \mathbb{N} \rightarrow \mathbb{N}, x \mapsto x^2$ und

```
fun f (x:int,y:int) :int = x*y
```

berechnet die Funktion $f: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}, (x, y) \mapsto x \cdot y$

- (a)

```
fun mul(a:int, n:int, x:int) : int =  
  if n=0  
  then a  
  else mul(a+x,n-1,x)
```
- (b)

```
fun quer(a:int, x:int) : int =  
  if x=0  
  then a  
  else quer(a + x mod 10,x div 10)
```
- (c)

```
fun rev(m:int, n:int) : int =  
  if n=0  
  then m  
  else rev(m*10 + n mod 10,n div 10)
```

Aufgabe 1.2

Stellen Sie das Ausführungsprotokoll für die folgenden Aufrufe der oben genannten Prozeduren auf:

- (a) $rev(538, 10)$
(b) $mul(17, 3, 10)$
(c) $quer(9273, 0)$

Hinweis: Überlegen Sie sich gegebenenfalls neu, was die obigen Prozeduren berechnen.

Aufgabe 1.3

Wie erkennt man endrekursive Prozeduren am Ausführungsprotokoll? Finden Sie mindestens zwei Charakterisierungen!

2 Sprache der Informatik

Aufgabe 1.4

Betrachten Sie das folgende Programm:

```
fun f (x:int) :int =
  if x=42
  then f(x-1) + 4
  else 10
fun g (x:real) :real =
  x * x + 2.0 * x + 1.0
val s = (f(1),g(1.0))
val t = (g(3.14), f(42))
fun g (x:real) : real =
  x * x * x + 3 * x * x + 3 * x + 1
```

Identifizieren Sie Schlüsselwörter, Konstanten, Bezeichner und Operatoren. Was tut das Programm? An was wird der Bezeichner s gebunden?

Aufgabe 1.5

Betrachten Sie die folgende Prozedurdeklaration:

```
fun f (x:int) :bool = f(x)
```

Identifizieren Sie den Prozedurtyp, den Argumenttyp und den Ergebnistyp der Prozedur. Was ist der Rumpf der Prozedur?

Hinweis: Ist die Prozedur wohlgetypt? Begründen Sie Ihre Antwort!

Aufgabe 1.6

Stellen Sie sich vor, Sie könnten eine Woche in die Vergangenheit reisen und treffen sich selbst. Sie kennen zu diesem Zeitpunkt das Konzept von Divergenz noch nicht. Wie würden Sie sich Divergenz erklären? Finden Sie ein anschauliches Beispiel!

3 Kleine Programme schreiben

Aufgabe 1.7

Schreiben Sie ein kleines Programm, das den Bezeichner s an den Wert 5 bindet. Verwenden Sie dabei den Wert 5 *nicht* explizit, d.h. die Konstante 5 darf in Ihrem Programm nicht vorkommen.

Aufgabe 1.8

Schreiben Sie ein Programm, in dem jeder Typ, den Sie bisher kennengelernt haben sowohl als Argumenttyp einer Prozedur als auch als Ergebnistyp einer Prozedur mindestens einmal vorkommt.

Aufgabe 1.9

Schreiben Sie ein Programm mit zwei Prozedurdeklarationen und fünf Prozeduranwendungen.

4 Rekursive Prozeduren schreiben

Aufgabe 1.10

Schreiben Sie eine rekursive Prozedur $fakultaet: int \rightarrow int$, die für eine natürliche Zahl n die Fakultät berechnet und für alle anderen Aufrufe divergiert. Geben Sie zunächst die Rekursionsgleichungen an.

Aufgabe 1.11

Geben Sie die Rekursionfolge und das Ausführungsprotokoll Ihrer Prozedur aus 1.10 für $fakultaet(3)$ an.

Aufgabe 1.12

Schreiben Sie eine nicht-rekursive Prozedur *fakultaet3*: $int \rightarrow int * int * int$ die für eine natürliche Zahl n die einen Tupel zurückgibt, der in der ersten Komponente die Fakultät von $(n - 1)$, in der zweiten Komponente die Fakultät von n und in der dritten Komponente die Fakultät von $(n + 1)$. Ist mindestens eine der Zahlen negativ, so ist es egal, was Ihre Prozedur zurückgibt.

Aufgabe 1.13

Erklären Sie anhand der endrekursiven Variante der Prozedur Fakultät aus Aufgabe 1.10 das Konzept des Akkus.

Aufgabe 1.14

Sie haben einige Beispiele für rekursive Prozeduren bereits im Buch, auf dem Übungsblatt und in der Vorlesung gesehen. Schreiben Sie für mindestens drei Beispiele die zugehörigen Rekursionsgleichungen auf. Wandeln Sie diese Rekursionsgleichungen dann in endrekursive um.

Überlegen Sie sich dann ein allgemeines Vorgehen wie Sie aus Rekursionsgleichungen SML-Prozeduren schreiben.

5 Knobelaufgaben

Aufgabe 1.15 (Zum Üben von Syntax)

Schreiben Sie in SML eine Prozedur *taschenrechner*, die natürliche Zahlen addieren, subtrahieren, dividieren und multiplizieren kann.

Hinweis: Überlegen Sie sich, wie man entscheiden kann, welche Operation ausgeführt werden soll.

Aufgabe 1.16 (Zu Rekursionsgleichungen)

Überlegen Sie sich nicht-endrekursive Rekursionsgleichungen für die Berechnung von $\lfloor \sqrt{n} \rfloor$ für eine natürliche Zahl n . Ist die ohne Probleme möglich? Was fällt Ihnen auf?

Aufgabe 1.17 (Verzwickte Prozedur)

Betrachten Sie den Anfang der folgenden Folge $(a_n)_{n \in \mathbb{N}}$ von Zahlen, die man auch *Tower of Twos* nennt:

$$a_0 = 2, a_1 = 2^2, a_2 = 2^{(2^2)}, a_3 = 2^{(2^{(2^2)})}, \dots$$

Schreiben Sie eine SML-Prozedur, die für eine natürliche Zahl n die n -te Zahl dieser Form a_n berechnet. Für welches $n \in \mathbb{N}$ können Sie diese Zahl noch berechnen? Mit welcher Exception wird die Berechnung abbrechen?