DEPENDABLE SYSTEMS AND SOFTWARE

Fachrichtung 6.2 — Informatik
Prof. Dr.-Ing. Holger Hermanns
Dipl.-Inform. Lijun Zhang

Übungsblatt 2 (Programmierung I)

Lesen Sie im Skript: Kapitel 2-3.2

Aufgabe 2.1: (Einfache Funktion)
Schreiben Sie eine Prozedur $p : real*real \to real$, die die Funktion $f(x, y) = (x-3)(y+5)^2$ mit Gleitkommazahlen berechnet. Verwenden Sie eine lokale Deklaration, damit die Addition $y + 5$ nur einmal berechnet werden muss.

Aufgabe 2.2: (Potenzfunktion)
Schreiben Sie eine rekursive Prozedur $power : real * int \to real$, die zu einer reellen Zahl x und einer natürlichen Zahl n die Potenz x^n mittels Gleitkommaoperationen berechnet. Welche Zahl liefert $power(3.0, 100)$? Handelt es sich dabei wirklich um die Zahl 3^{100}?

Aufgabe 2.3: (Newtonsches Verfahren)
Schreiben Sie eine Prozedur $sqrt : real \to real$, die zu x die erste Approximation a_n liefert, die $\sqrt{x}$ mit der Genauigkeit $|x - a_n^2| < 10^{-4}$ approximiert. Schreiben Sie zusätzlich eine Prozedur $sqrt' : real \to real * int$, die zu x das Paar (a_n, n) liefert, damit Sie sehen können, wieviele Approximationsschritte jeweils erforderlich sind.

Aufgabe 2.4: (Baumdarstellung)
Geben Sie die Baumdarstellungen der folgenden durch Zeichendarstellungen beschriebenen Phrasen an.

(a) `x + 3 * f x - 4`

(b) `1 + 2 + 3 + 4`

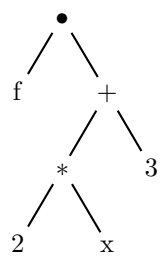
(c) `1 + 2 * x - y * 3 + 4`

(d) `int * (int * int) * int -> real`

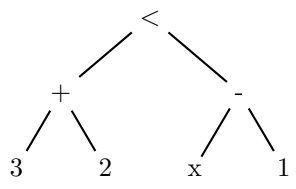
(e) `fun p(x: int, n: int): int = if n > 0 then x * p(x, n - 1) else 1`

Aufgabe 2.5: (Minimal geklammerte Zeichendarstellung)
Geben Sie für die folgenden Ausdrücke minimal geklammerte Zeichendarstellungen an:

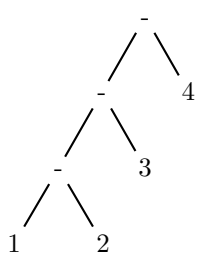
(a)



(b)



(c)



Sie können die möglichen Zeichendarstellungen durch gezieltes Experimentieren mit einem Interpreter ermittlen.

Aufgabe 2.6: (Bindungen)
Welche Bindungen berechnet das folgende Programm?

```
fun f (x: bool) = if x then 1 else 0
val x = 5 * 7
fun g (z:int) = f(z < x) < x
val x = g 5
```

Aufgabe 2.7: (Prozeduren)
Geben Sie einen geschlossenen Ausdruck an, der eine Prozedur $int \to int$ beschreibt, die zu x das Ergebnis x^2 liefert. Geben Sie die Tripeldarstellung der durch Ihren Ausdruck beschriebenen Prozedur an. Hinweis: Verwenden Sie einen Let-Ausdruck.

Aufgabe 2.8: (Min)
Deklarieren Sie eine Prozedur $min : int \to int \to int \to int$, die zu 3 Zahlen die kleinste liefert. Deklarieren Sie min auf 3 Arten: Mit einer kaskadierten Prozedurdeklaration, mit einer Prozedurdeklaration und zwei Abstraktionen, und mit einer Deklaration mit val und drei Abstraktionen.

Aufgabe 2.9: (Baumdarstellung)
Geben Sie die Baumdarstellung des Ausdrucks

```
plus x y + plus x (y + 2) * 5
```

an. Überprüfen Sie dir Richtigkeit Ihrer Darstellung mit einem Interpreter.

Aufgabe 2.10: (Cas und Car)
Schreiben Sie zwei Prozeduren

$$cas : (int * int \to int) \to int \to int \to int$$
$$car : (int \to int \to int) \to int * int \to int$$

sodass cas zur kartesischen Darstellung einer zweistelligen Operation die kaskadierte Darstellung und car zur kaskadierten Darstellung die kartesische Darstellung liefert. Erproben Sie cas und car mit Prozeduren, die das Maximum zweier Zahlen liefern:

```
fun maxCas (x:int) (y:int) = if x<y then y else x
fun maxCar (x:int, y:int) = if x<y then y else x
val maxCas' = cas maxCar
val maxCar' = car maxCas
```

Wenn Sie cas und car richtig geschrieben haben, verhält sich $maxCas'$ genauso wie $maxCas$ und $maxCar'$ genauso wie $maxCar$.
Die Lösung der Aufgabe ist sehr einfach, experimentieren Sie mit einem Interpreter!

Aufgabe 2.11: (Prozedurdarstellung)
Geben Sie die Tripeldarstellung der Prozedur an, zu der der folgende Ausdruck auswertet:

```
let val a = 7
    fun f (x:int) = a + x
    fun g (x:int) (y:int): int = g (f x) y
in
    g (f 5)
end
```

Aufgabe 2.12: (Mul ohne Rekursion)
Deklarieren Sie eine Prozedur $mul : int \to int \to int$, die das Produkt zweier Zahlen x und n gemäß der Gleichung

$$x \cdot n = 0 + \underbrace{x + ... + x}_{n-mal}$$

durch Addieren berechnet ($n \geq 0$). Die Prozedur mul soll mithilfe der Prozedur $iter$ formuliert werden und nicht rekursiv sein.