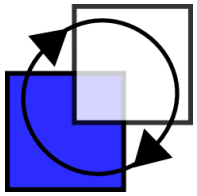
REACTIVE SYSTEMS GROUP

Universität des Saarlandes

Prof. Bernd Finkbeiner, Ph.D.
Markus Rabe, M.Sc.



Programmierung 1 (SS 2010) - 3. Übungsblatt

http://react.cs.uni-saarland.de/prog1/

Lesen Sie im Buch bis Kapitel 3.1

Bei Problemen gehen Sie in die Office Hours: Mo-Mi von 13 bis 14 Uhr.

Aufgabe 3.1 (Zum wach werden)

Schreiben Sie eine Prozedur $mymod : int \to int \to int$, die für $x \geq 0$ und $y \geq 1$ dasselbe Ergebnis wie $x \ mod \ y$ liefert. Geben Sie dazu zunächst Rekursionsgleichungen für $x \ mod \ y$ an.

Aufgabe 3.2 (Wiederholung)

Schreiben Sie eine rekursive Prozedur $hoch : int * int \to int$, die zu einer Zahl x und einer zweiten Zahl y die Potenz x^y berechnet. Schreiben Sie die Prozedur einmal endrekursiv und einmal nicht endrekursiv.

Aufgabe 3.3 (Divergenz und Ausführungsreihenfolge)

Betrachten Sie die folgenden Deklarationen und erklären Sie, warum sich die Prozeduren p und r nicht für alle Argumente gleich verhalten.

```
fun p (n:int) : int = if n=0 then n else p(n-1)
fun q (b:bool, x:int, y:int) = if b then x else y
fun r (n:int) : int = q(n=0, n, p(n-1))
```

Aufgabe 3.4

Schreiben Sie eine Prozedur $quer : int \to int$, die die Quersumme einer ganzen Zahl mithilfe einer endrekursiven Hilfsprozedur berechnet.

Aufgabe 3.5 (Üben üben üben)

Schreiben Sie eine Prozedur $rev' : int * int \to int$, sodass $rev'(0, x)$ für jede natürliche Zahl x die Reversion von x liefert. Verwenden Sie dabei keine Hilfsprozedur. Machen Sie sich klar, dass rev' Zahlen mithilfe eines Akkus reversiert.

Aufgabe 3.6

Betrachten Sie das folgende Programm:

```
fun f x = if x = 0 then 1 else 15*f(x-1)+2
```

a) Geben Sie den Typen der Prozedur an.

b) Deklarieren sie eine semantisch äquivalente endrekursive Prozedur `f'`. Hinweis: Benutzen sie einen Akkumulator und eine Hilfsprozedur.

Aufgabe 3.7 (Baumdarstellung)

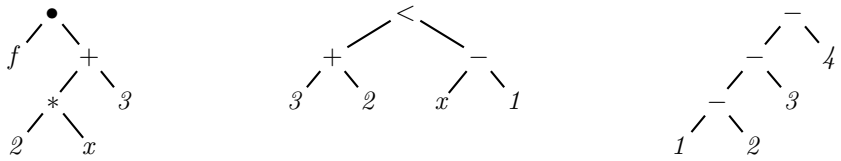
Geben Sie die Baumdarstellungen der folgenden durch Zeichendarstellungen beschriebenen Phrasen an.

```
x+3*f x-4
1+2+3+4
1+2 * x-y * 3+4
int * (int * int) * int -> real
fun p(x:int,n:int):int=if n>0then x*p(x,n-1)else 1
```

Überprüfen Sie Ihre Baumdarstellungen mit einem Interpreter.

Aufgabe 3.8 (Zeichendarstellung)

Geben Sie für die folgenden Ausdrücke minimal geklammerte Zeichendarstellungen an:



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

Aufgabe 3.9

Geben Sie die Tripeldarstellung der Prozedur an, zu der der folgende Ausdruck auswertet:

```
let
   val a = 1
   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 3.10

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 und lesen Sie sich die Begrifflichkeiten im Buch genau durch.

Aufgabe 3.11

Seien U und V zwei Umgebungen. Unter welchen Bedingungen über V und U gilt

$$U + V = V + U \ ?$$

Aufgabe 3.12

Entscheiden Sie jeweils, ob folgende Programme semantisch äquivalent sind:

a)	`fun a (x:int) = x + a`	`fun a(x:int) = a + x`

b)	`fun fac (x:int) =`
	`  if x = 0 then 1`
	`  else fac(x-1)`
	`fun b (p:int*int) = (#2p,#1p)`
	`val it = b(fac(~1),fac(1))`
		`fun fac (x:int) =`
		`  if x = 0 then 1`
		`  else x * fac(x-1)`
		`fun b(p:int*int) = p`
		`val it = b(fac(1),fac(~1))`

c)	`val it = 111111111111 mod 1`	`val it = 111111 mod 1`
d)	`fun di (x:int) = di (x-1)`	`fun di (x:int) = 1 + di (x-1)`

Aufgabe 3.13 (Freie Bezeichner)

Bestimmen Sie die freien und gebundenen Bezeichner in folgendem Ausdruck:

```
5 + x * (fn x:int => x + y) y
```

Aufgabe 3.14 (Ausblick)

Schreiben Sie eine zu der folgenden Prozedur `f` semantisch äquivalente Prozedur ohne dabei das Schlüsselwort `fun` zu verwenden.

```
fun f x = fn y => x*y
```