



Programmierung WS 2002 / 03: Musterlösung zum 1. Übungsblatt

Prof. Dr. Gert Smolka, Dipl.-Inform. Thorsten Brunklaus

Aufgabe 1.1: Signum (5)

```
fun signum(x:int) =  
  if x > 0 then 1  
  else if x < 0 then ~1  
  else 0
```

Aufgabe 1.2: 17. Potenz (8)

```
fun q (x:int)      = x * x  
fun hoch17 (x:int) = x * q (q (q x))
```

Aufgabe 1.3: Gleitkomma-Potenzen (8)

```
fun power (x:real, n:int) : real =  
  if n<1 then 1.0 else x*power(x,n-1)
```

`power(3.0,100)` liefert die Zahl `5.15377520732E47`. Dies ist ungleich 3^{100} .

Aufgabe 1.4: Fakultäten (10)

```
fun fak (n:int) : int = if n=0 then 1 else n*fak(n-1)
```

Für $n = 12$ erhält man den Wert `479001600`, für $n > 12$ wird die Ausführung wegen Überlaufs abgebrochen.

Aufgabe 1.5: Größte gemeinsame Teiler (10)

```
fun ggt(x:int, y:int) : int =  
  if x = y then x  
  else if x > y then ggt(x - y, y)  
  else ggt(x, y - x)
```

Aufgabe 1.6: Quersumme (10)

```
fun quer(x:int) : int =  
  if x<0 then quer(~x)  
  else if x=0 then 0  
  else (x mod 10) + quer(x div 10)
```

Aufgabe 1.7: Natürliche Quadratwurzel (10)

```
fun wurzel'(n:int, x:int) : int = if n*n>x then n else wurzel'(n+1,x)  
  
fun wurzel(x:int) = wurzel'(0,x) - 1
```

Aufgabe 1.8: Standardstruktur Math (5)

```
fun g(x:real) = 2.0 * x + 1.4
fun h(x:real) = Math.sin x + Math.cos(2.0 * Math.pi * x)
```

Aufgabe 1.9: Umgebungen (10)

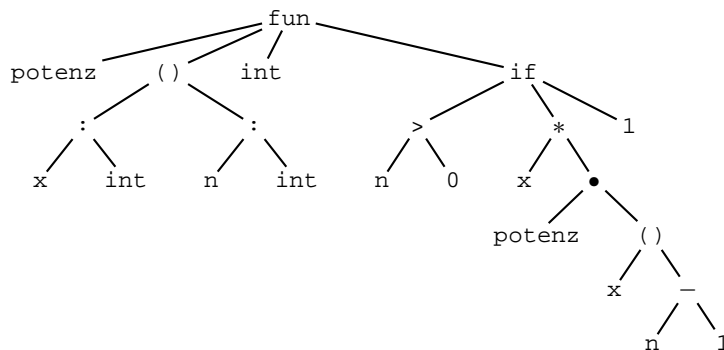
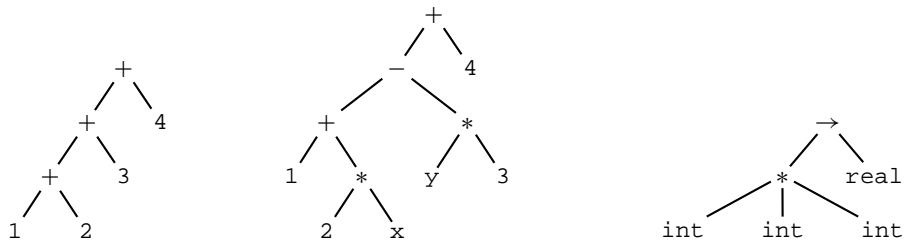
```
{ f ↦ (fun f(x:bool) = if x then 1 else 0, {}),
  y ↦ 36,
  g ↦ (fun g(z:int) = f(z<y)<x,
       { f ↦ (fun f(x:bool) = if x then 1 else 0, {}), y ↦ 36, x ↦ 35 }) ,
  x ↦ true }
```

Aufgabe 1.10: Klammersparregeln (6)

```
f (2 * x + 3)
f 2 * (x + 3)
f 2 * x + 3
```

Aufgabe 1.11: Baumdarstellungen (8)

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



Aufgabe 1.12: Auswertungsprotokolle (10)

```
f(3,1)
-> if 3=0 then 1 else f(3-1, 1*3)
-> if false then 1 else f(3-1, 1*3)
-> f(3-1, 1*3)
-> f(2, 1*3)
-> f(2,3)
-> if 2=0 then 3 else f(2-1, 3*2)
-> if false then 3 else f(2-1, 3*2)
-> f(2-1, 3*2)
-> f(1, 3*2)
-> f(1, 6)
-> if 1=0 then 6 else f(1-1, 6*1)
-> if false then 6 else f(1-1, 6*1)
-> f(1-1, 6*1)
-> f(0, 6*1)
-> f(0, 6)
-> if 0=0 then 6 else f(0-1, 6*0)
-> if true then 6 else f(0-1, 6*0)
-> 6
```