



## Programmierung: Musterlösung zum 1. Übungsblatt

Thorsten Brunklaus

### Aufgabe 1.1: Prozedurdeklarationen (3+3)

```
fun g(x:real) = 2.0 * x + 1.4
```

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

### Aufgabe 1.2: Let-Ausdruck (7)

```
fun f(x : real, y : real) =  
  let  
    val z = y + 5.0  
  in  
    (x - 3.0) * z * z  
  end
```

### Aufgabe 1.3: Signum (7)

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

### Aufgabe 1.4: Fakultät (6+4)

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

Das grösste  $n$ , für welches Moscow ML die Fakultät noch darstellen kann, ist 12:  $12! = 479001600$  (für die Darstellung von  $13! = 6227020800$  benötigt man mehr als 32 Bit).

### Aufgabe 1.5: Binomialkoeffizient (10)

```
fun binom(n : int, k : int) =  
  if k = 0 then 1  
  else  
    if n = 0 then 0  
    else n * binom(n - 1, k - 1) div k
```

Für negative  $n, k$  war die Funktion nicht spezifiziert, deshalb kann das Ergebnis in diesen Fällen beliebig sein.

### Aufgabe 1.6: Quersumme (10)

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

### Aufgabe 1.7: N-te Primzahl (Challenge, ohne Bewertung)

```
fun primeIter(i : int , p : int) =
  if ((i * i) <= p) then
    if ((p mod i) = 0) then false else primeIter((i + 2), p)
  else
    true

fun isPrime(n : int) =
  if ((n mod 2) = 0) then false
  else primeIter(3, n)

fun doCheckPrime(p : int, n : int) =
  if isPrime(p) then
    if (n = 1) then p else doCheckPrime((p + 2), (n - 1))
  else
    doCheckPrime((p + 2), n)

fun nthPrime(n : int) =
  if (n = 1) then 2 else doCheckPrime(3, (n - 1))
```