Alexander Blotny, Stefan Kadereit
Tutor: Till Zoppke, Gruppe 1

Aufgabe 1

```
f :: Int -> Int -> Int
f a b = b * div a b

g :: Int -> Int -> Int -> Bool
g a b c = a <= b^c
```

a) `f x :: Int -> Int`
`b ↦ b * div x b`
Das Ergebnis der Ganzzahldivision von x und b wird mit b multipliziert. Für alle x kleiner als b ergibt sich der Wert 0 für die Ganzzahldivision, also auch für die Funktion.

b) `g x :: Int -> Int -> Bool`
`(b,c) ↦ x <= b^c`
Diese Funktion nimmt den Wert true an, wenn x kleiner ist als b^c. Falls b den Wert 1 annimmt oder c den Wert 0 besitzt, dann muss x kleiner oder gleich 1 sein.

c) `g x y :: Int -> Bool`
`c ↦ x <= y^c`
Für jedes x kleiner gleich y^c wird diese Funktion wahr.

d) `g (f x y) z :: Int -> Bool`
`c ↦ (y * div x y) <= z^c`
Für jedes $x \in \mathbb{N} \wedge x < y$ kann c jede natürliche Zahl sein, damit die Funktion wahr wird.

Aufgabe 2

a) Mit Hilfe des Restes der Ganzzahldivision wird bestimmt, ob x ein Teiler von y ist.

```
(!!!)::Int->Int->Bool
x !!! y
 |x==0 = error "0 kann kein Teiler sein" -- Abbruch bei ungültiger Eingabe
 |mod y x == 0 = True -- x ist Teiler von y, da Rest der Ganzzahldivision 0 ist
 |otherwise = False -- x ist kein Teiler, da Rest der Ganzzahldivision /= 0 ist
```

b) Anhand des gegebenen Rekursionsschemas wird mit der Rekursion der Binominalkoeffizient bestimmt.

```
(#) :: Int->Int->Int
n # k
 |k==0 = 1 -- Rekursionsverankerung nach Definition des Rekursionsschemas
 |n==0 = 0 -- Rekursionsverankerung nach Definition des Rekursionsschemas
 |otherwise = ((n-1)#k) + ((n-1)#(k-1))
                        -- Rekursion (Wiederholung bis n==0 oder k==0)
```

Aufgabe 3

a) Die Formel zur Berechnung des Volumens wird benutzt.

```
volumen :: Float->Float->Float
volumen r h = (pi * r^2 * h) / 3 -- Errechnung des Volumens
```

b) Die Lösung der Gleichung wird mit der Formel $x_{1/2} = -\frac{b}{2a} \pm \sqrt{\frac{b^2}{4a^2} - \frac{c}{a}}$ berechnet. Dabei werden zunächst die Spezialfälle behandelt, um u.a. eine Division durch 0 zu vermeiden.

```
loesungen :: Float->Float->Float->(Float,Float)
loesungen a b c
 |a==0 && b==0 && c==0 = error "alle reellen Zahlen sind Loesung"
-- Für x alle reellen Zahlen einsetzbar, da sich die Gleichung 0=0 ergibt
 |((a==0) && (b==0)) = error "keine Loesungen"
-- In der Gleichung entsteht eine falsche Aussage (z.B. 5=0, für c=5)
-- c kann nicht 0 sein, da in dem Fall schon vorher abgebrochen wurde
 |(a==0) = (-(c)/b,-(c)/b)
-- Für a=0 entsteht eine Gerade, die nur einen Schnittpunkt mit der x-Achse hat
 |((b^2)/(4*a^2)-(c/a)<0) = error "keine Loesungen"
-- In der Lösungsformel entsteht ein negativer Wert unter der Wurzel
 |otherwise = ((-(b/(2*a))+sqrt((b^2)/(4*a^2)-(c/a))),(-(b/(2*a))-sqrt((b^2)/(4*a^2)-(c/a))))
-- Da alle Spezialfälle behandelt wurden, kann Lösungsformel angewandt werden
```

Testläufe

```
Main> 3 !!! 6
True
Main> 6 !!! 3
False
Main> 5 !!! 25
True
Main> 0 !!! 5
Program error: 0 kann kein Teiler sein
Main> 5 !!! 0
True

Main> 6 # 2
15
Main> 0 # 0
1
Main> 3 # 4
0
Main> 10 # 5
252
Main> 2 # 1
2

Main> volumen 3 4
37.6991
Main> volumen 0 1
0.0
Main> volumen 2 3
12.5664
Main> volumen 5 2
52.3599
Main> volumen 10 20
2094.4

Main> loesungen (-1) 5 2
(5.37228,-0.372281)
Main> loesungen 1 5 2
(-0.438447,-4.56155)
Main> loesungen 1 6 5
(-1.0,-5.0)
Main> loesungen 0 7 9
(-1.28571,-1.28571)
Main> loesungen 0 3 6
(-2.0,-2.0)
Main> loesungen 0 0 6
Program error: keine Loesungen
Main> loesungen 0 0 0
Program error: alle reellen Zahlen sind Loesung
Main> loesungen 1 2 3
Program error: keine Loesungen
Main> loesungen 0 0 3
Program error: keine Loesungen
Main> loesungen 4 9 3
(-17.2819,-18.7181)
```