(-) ((+) ((+) 1 2) 3) (-2) => (-) ((+) 3 3) (-2) => (-) 6 (-2) => 8

2\*\*3 + (2^3) => 8.0 + 8 = 16.0

log 1 => 0.0 // e^0.0 = 1

div 1 2.0 => Fehler // div kann nur für ganze Zahlen verwendet werden

mod 4 (-3) => -2 //das Rest

(-4 `mod` 5) == (-4 `rem` 5) => True //-4 == -4

(4 `mod` (-5)) == (4 `rem` (-5)) => False // -1 == 4

succ 4 \* 8.0 => 5 \* 8.0 = 40.0 //succ gibt das nächste in eine Folge

succ (4 \* 8) => succ 32 => 33

if (mod 1 2)==0 then "ja" else "nein" => “nein” // weil mod 1 2 => 1

True || undefined => True //bei “ODER” muss nur eine Aussage gelten. Wenn die erste gilt, wird die

Zweite gar nicht betrachtet. (lazy evaluation)

True && not (True || undefined) => True && not(True) => True && False => False

True && (undefined || True) => Fehler “Exception: Prelude.undefined”, weil undefined kein Bool ist,

steht erst in “ODER”

3/0 \*\* 2 => 3/0 => Infinity

2 - 0/0 => 2 – NaN => NaN