WiSe 2015/2016

**Funktionale Programmierung**

**Übungsblatt** (Abgabe: Mo., den 26.10. um 10:10 Uhr)

Author: Boyan Hristov Matrikelnummer: 4980301

1. Aufgabe
2. div 1 2 => 0, denn 1/2 = 0.5, aber div wird für ganzzahlige Teilungen verwendet und immer nach unten gerundet
3. 8^0 => 1, denn x^0 immer 1 ist
4. rem 5 2 => 1 denn rem das Rest nach einer Teilung kalkuliert
5. sqrt 3 => 1.7320508075688772 (Würzel aus drei ist 1.732)
6. "ba"<"aba" => False, weil ‘b’ nach ‘a’ alphabetisch steht
7. 8^200 => 41495155688809929585124639970704894710379428181803401509523685376   
   (8 hoch 200)
8. 4 /= 5 => True, weil 4 nicht gleich 5 ist
9. mod 5 2 => 1, weil das Rest von 5/2 1 ist
10. sqrt (-1) => NaN (Not a Number), weil kein Würzel von eine negative Zahl gibt
11. True < False => False, weil (ich vermute) True als 1 gelten kann und False als 0. Man kann aber nicht Bool mit Zahl vergleichen, deswegen... o.O
12. 0.1== 0.3/3 => False, weil 0.1 im Speicher nicht genau vorstellbar ist
13. rem 5 (-2) => 1, weil das positive Rest von 5/(-2) gleich 1 ist
14. exp 1 => e^1 = 2.718281828459045
15. True || False => True, weil bei “ODER” muss nur eine Aussage wahr sein
16. 2\*\*1024 => Infinity – eigentlich ist das 2^1024, aber \*\* ist ^ für nicht ganze Zahle, wobei nicht genug Speicher gibt (?)
17. abs -7 => Fehler, weil es abs (-7) sein muss. So wäre es 7 sein, weil abs immer eine positive Zahl gibt
18. mod 5 (-2) => -1
19. 'z'<'A' => False, weil ASCII Code von ‘z’ 122 ist und von ‘A‘ 65
20. True && False => False, weil bei “AND“ müssen beide Aussagn wahr sein
21. Aufgabe
22. (-) ((+) ((+) 1 2) 3) (-2) => (-) ((+) 3 3) (-2) => (-) 6 (-2) => 8
23. 2\*\*3 + (2^3) => 8.0 + 8 = 16.0
24. log 1 => 0.0 // e^0.0 = 1
25. div 1 2.0 => Fehler // div kann nur für ganze Zahlen verwendet werden
26. mod 4 (-3) => -2 //das Rest
27. (-4 `mod` 5) == (-4 `rem` 5) => True //-4 == -4
28. (4 `mod` (-5)) == (4 `rem` (-5)) => False // -1 == 4
29. succ 4 \* 8.0 => 5 \* 8.0 = 40.0 //succ gibt das nächste in eine Folge
30. succ (4 \* 8) => succ 32 => 33
31. if (mod 1 2)==0 then "ja" else "nein" => “nein” // weil mod 1 2 => 1
32. True || undefined => True //bei “ODER” muss nur eine Aussage gelten. Wenn   
     die erste gilt, wird die
33. Zweite gar nicht betrachtet. (lazy evaluation)
34. True && not (True || undefined) => True && not(True) => True && False =>   
     False
35. True && (undefined || True) => Fehler “Exception: Prelude.undefined”, weil   
     undefined kein Bool ist, steht erst in “ODER”
36. 3/0 \*\* 2 => 3/0 => Infinity
37. 2 - 0/0 => 2 – NaN => NaN