3a) i) (\x y → x+2*y) (2*6) (3*8)
evaluiert laut ghci zu 60 ✓ 3/3

ii) (\f g h → f (g (x)) (g (x)))
(\x y → x + 2*y)
((\x y z → ((\x y → x + 2*z) x z)
-((\x y → x + 2*y) (2*x) y))
1 2) 3
evaluiert laut ghci zu 3 ✓ 5/5

iii) (\x y → x + 2*y)
((\x y z → ((\a b → a+2*b) x z)
-((\a b → a+2*b) (2*x) y)) 1 2 3)
((\x y → x + 2*y) 2 7)
evaluiert laut ghci zu 33 ✓ 5/5

∑ = 45/50

b) Wenn man Data.Function importiert, ist es problemlos möglich die Collatz-Funktion mit Lambda auszudrücken aus zu drücken:

```
import Data.Function
fix (\c n → if n == 1 then [1]
      else if even n then n : c (n `div` 2)
      else n : c (3*n + 1)) 5
```

evaluiert zu [5,16,8,4,2,1] laut ghci
Andernfalls, ohne den der Fixpunktoperation, würde sich der Aufruf der Lambda-Funktion für jede Rekursion vergrößern, mit stets anderen Werten für n.
Sollte die Collatzfunktion für eine Eingabe nicht terminieren, so wäre der Funktionsaufruf unendlich lang. Das geht nicht!
✓ 2/2

3/5

1a) $S := (\lambda x.\lambda y.\lambda z((x\,y)\,z))(\lambda x.\lambda y(x(\lambda y.\lambda z.z))$

b) $(\backslash x\,y\,z \to x\,y\,z)(\backslash x\,y \to (x)(\backslash y\,z \to z))$ Ausgabe? 2/5

c) Alle Variablen sind gebunden.

$\Rightarrow (\backslash abc \to abc)(\backslash de \to d)(\backslash fg \to g)$ 5/5

a) $S := (\lambda x.\lambda y.\lambda z((x\,y)\,z))(\lambda x.\lambda y\,(x\,(\lambda y.(\lambda z\,z))))$

b) $(\backslash x\,y\,z \to x\,y\,z)(\backslash x\,y \to x(\backslash y\,z \to z))$

No instance Show $(((t_2 \to t_3 \to t_3) \to t_0) \to t_1 \to t_0)$

2) collatz 1 = trace "1" 1

collatz n = trace show $n if even n then collatz (n `div` 2)

else collatz (3*n+1)

2)

Haskell Code:

```
"
import System.IO
import Data.Char

main = do

 writeFile "01.txt" (show $ collatz $ 2^33)
 writeFile "02.txt" (show $ collatz $ 235)
 writeFile "03.txt" (show $ collatz $ 5)
 writeFile "04.txt" (show $ collatz $ 82654)
 writeFile "05.txt" (show $ collatz $ 1337)
 writeFile "06.txt" (show $ collatz $ 420)
 writeFile "07.txt" (show $ collatz $ 69)
 writeFile "08.txt" (show $ collatz $ 66)
 writeFile "09.txt" (show $ collatz $ 42)
 writeFile "10.txt" (show $ collatz $ 31415926)

collatz :: Integer -> [Integer]
collatz a = if a == 1 then [1] else
  if even a then a : collatz (a `div` 2)
  else a : collatz ((3*a)+1)
"
```

braucht man wirklich System.IO ?

Die Idee ist richtig trotzdem solltest du, wenn du vorhast in Haskell zu programieren "trace" lernen

20/20

Erzeugt Ausgaben

01.txt
[8589934592,4294967296,2147483648,1073741824,536870912,268435456,1342177
28,67108864,33554432,16777216,8388608,4194304,2097152,1048576,524288,2621
44,131072,65536,32768,16384,8192,4096,2048,1024,512,256,128,64,32,16,8,4,2,1]

02.txt
[235,706,353,1060,530,265,796,398,199,598,299,898,449,1348,674,337,1012,506,25
3,760,380,190,95,286,143,430,215,646,323,970,485,1456,728,364,182,91,274,137,4
12,206,103,310,155,466,233,700,350,175,526,263,790,395,1186,593,1780,890,445,1
336,668,334,167,502,251,754,377,1132,566,283,850,425,1276,638,319,958,479,143
8,719,2158,1079,3238,1619,4858,2429,7288,3644,1822,911,2734,1367,4102,2051,6
154,3077,9232,4616,2308,1154,577,1732,866,433,1300,650,325,976,488,244,122,61
,184,92,46,23,70,35,106,53,160,80,40,20,10,5,16,8,4,2,1]

03.txt
[5,16,8,4,2,1]

04.txt
[82654,41327,123982,61991,185974,92987,278962,139481,418444,209222,104611,
313834,156917,470752,235376,117688,58844,29422,14711,44134,22067,66202,331

01,99304,49652,24826,12413,37240,18620,9310,4655,13966,6983,20950,10475,314
26,15713,47140,23570,11785,35356,17678,8839,26518,13259,39778,19889,59668,2
9834,14917,44752,22376,11188,5594,2797,8392,4196,2098,1049,3148,1574,787,23
62,1181,3544,1772,886,443,1330,665,1996,998,499,1498,749,2248,1124,562,281,84
4,422,211,634,317,952,476,238,119,358,179,538,269,808,404,202,101,304,152,76,3
8,19,58,29,88,44,22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1]

05.txt
[1337,4012,2006,1003,3010,1505,4516,2258,1129,3388,1694,847,2542,1271,3814,1
907,5722,2861,8584,4292,2146,1073,3220,1610,805,2416,1208,604,302,151,454,22
7,682,341,1024,512,256,128,64,32,16,8,4,2,1]

06.txt
[420,210,105,316,158,79,238,119,358,179,538,269,808,404,202,101,304,152,76,38,1
9,58,29,88,44,22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1]

07.txt
[69,208,104,52,26,13,40,20,10,5,16,8,4,2,1]

08.txt
[66,33,100,50,25,76,38,19,58,29,88,44,22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1]

09.txt
[42,21,64,32,16,8,4,2,1]

10.txt
[31415926,15707963,47123890,23561945,70685836,35342918,17671459,53014378,
26507189,79521568,39760784,19880392,9940196,4970098,2485049,7455148,3727
574,1863787,5591362,2795681,8387044,4193522,2096761,6290284,3145142,15725
71,4717714,2358857,7076572,3538286,1769143,5307430,2653715,7961146,398057
3,11941720,5970860,2985430,1492715,4478146,2239073,6717220,3358610,167930
5,5037916,2518958,1259479,3778438,1889219,5667658,2833829,8501488,4250744
,2125372,1062686,531343,1594030,797015,2391046,1195523,3586570,1793285,53
79856,2689928,1344964,672482,336241,1008724,504362,252181,756544,378272,1
89136,94568,47284,23642,11821,35464,17732,8866,4433,13300,6650,3325,9976,49
88,2494,1247,3742,1871,5614,2807,8422,4211,12634,6317,18952,9476,4738,2369,7
108,3554,1777,5332,2666,1333,4000,2000,1000,500,250,125,376,188,94,47,142,71,
214,107,322,161,484,242,121,364,182,91,274,137,412,206,103,310,155,466,233,700
,350,175,526,263,790,395,1186,593,1780,890,445,1336,668,334,167,502,251,754,37
7,1132,566,283,850,425,1276,638,319,958,479,1438,719,2158,1079,3238,1619,4858
,2429,7288,3644,1822,911,2734,1367,4102,2051,6154,3077,9232,4616,2308,1154,5
77,1732,866,433,1300,650,325,976,488,244,122,61,184,92,46,23,70,35,106,53,160,8
0,40,20,10,5,16,8,4,2,1]