[+1, +2, +3] 1 [2, 3, 4]

$\boxed{1}$ 1. func$_1$ :: [a → b] → a → [b]

func$_1$ fs a = map ($a) fs

2. func$_2$ :: (a → b) → Tree a → [b]

helper function: flatten :: Tree a → [a]

flatten ~~Node~~ Leaf = [ ]
flatten t1 x t2 =
  flatten t$_1$ ++ [x] ++ flatten t$_2$

func$_2$ f t = map f (flatten t)

3. ~~Ord a~~
func$_3$ :: Ord a ⇒ [a] → [a]

func$_3$ = sort

4. func$_4$ :: (a → b → c) → (a → b) → a → c

func$_4$ g f a = g a (f a)

5. $func_5 :: ((a \to b) \to c \to d) \to (a \to c \to b) \to c$
$\to d$

$\to c \to$

~~applySnd ::~~ $(a \to c \to b) \to (a \to b)$

~~applySnd f~~ c

$func_3\ g\ f\ c = g\ (flip\ f\ c)\ c$

$\boxed{3}$

1. Int $\cong$ 2·Int (четные)

to = ·2
from = /2

2. data Tree a = Leaf | Node (Tree a)
a
(Tree a)

$\cong$ data TreeNew a = Leaf | Node a
(Tree a)
(Tree a)

to Leaf = Leaf
to Tree(t_1 x t_2) = TreeNew (x t_1 t_2)
from Leaf = Leaf
from TreeNew (x t_1 t_2) = Tree(t_1 x t_2)

3. data List a = Nil | Cons a (List a)

$\cong$ data Tsil a = Lin | Sonc (Tsil a) a

to Nil = Lin
to Cons x xs = Sonc (to xs) x

from Lin = Nil

from Sonc xs x = Cons x (from xs)

E_1

4. Either a (Either b (c,d))

E_2

$\cong$ ~~(a,b,c,d)~~ Either (Either b (c,d)) a

to Left a = Right_2 a

to Right b = Left_2 b

from Left_2 b = Right b

from Right_2 a = Left a

5. $(a \to b, a \to c) \cong (a \to c, a \to b)$

to = swap

from = swap

$\boxed{2}$ $Y = \lambda f. (\lambda x. f(x x)) (\lambda x. f(x x))$

$Y_3 = \lambda f. (\lambda x. f(x x x)) (\lambda x. f(x x x))$
$(\lambda x. f(x x x))$

$Y_4 = \lambda f. (\lambda x. f(x x x x x)) (\lambda x. f(x x x x))$
$(\lambda x. f(x x x x)) (\lambda x. f(x x x x))$

These are just other flavours
of Y- combinator, that
can't be reduced