```haskell
import System.IO

-- Lambda kalkul
{-

Pro vyraz E je pevny bod k: E k = k
Operator pevneho bodu Y: Y E = k = E (Y E)

LET G = \ f x y. iszero y ? x : f (add x y)
LET SUM = Y G

SUM 2 3 0 =
  (Y G) 2 3 0 =
  (G (Y G)) 2 3 0 =
  (G SUM) 2 3 0 =
  ((\ f x y. iszero y ? x : f (add x y )) SUM) 2 3 0 =
  (iszero 3 ? 2 : SUM (add 2 3)) 0 =
  (SUM 5) 0 = SUM 5 0 =
  (Y G) 5 0 =
  (G (Y G)) 5 0 =
  (G SUM) 5 0 =
  ((\ f x y. iszero y ? x : f (add x y )) SUM) 5 0 =
  iszero 0 ? 5 : SUM (add 5 0) =
  5

-}

-- Dukaz

{-
Ukazat:
  foldr f a as = foA f as a

(.) f g y - f (g y)                     /1
id x = x                                /2

foA f [] = id                           /3
foA f (x:xs) = (f x) . (foA f xs)       /4

foldr _ a [] = a                        /5
foldr f a (x:xs) = f x (foldr f a xs)   /6

1)
  as = []
  L = foldr f a []  =|5
    = a
  P = foA f [] a  =|3
    = id a  =|2
    = a
  L = P

I.H.
  foldr f a xs = foA f xs a

2) as = (x:xs)
  L = foldr f a (x:xs)  =|6
    = f x (foldr f a xs)  =|IH
    = f x (foA f xs a)
  P = foA f (x:xs) a  =|4
    = ((f x) . (foA f xs)) a  =|1
    = (f x) ((foA f xs) a)  =|priorita
    = f x (foA f xs a)
  L = P

Q.E.D.

-}


flns :: FilePath -> FilePath -> IO ()
flns fin fout = do
  hIn <- openFile fin ReadMode
  hOut <- openFile fout WriteMode
  c <- hGetContents hIn
  hPutStr hOut $ unlines $ f 1 $ lines c
  hClose hOut
  hClose hIn

f :: Int -> [String] -> [String]
f _ [] = []
f n (l:ls) =
  ((mk n)++" "++l) : f (n+1) ls

mk :: Int -> String
mk n = let
  sn = show n
  in sn++[' '| _<-[1..(3-length sn)]]


-- Premie

data Dbl a
    = Val a (Dbl a) (Dbl a)
    | Nil
    deriving (Show)

takeR _ Nil = []
takeR n (Val x _ r) =
  if n<1 then [] else x : takeR (n-1) r

tv = v1
  where
    v1 = Val 1 v4 v2
    v2 = Val 2 v1 v3
    v3 = Val 3 v2 v4
    v4 = Val 4 v3 v1
```