```haskell
import IO

{-

LET 0 = \ f n . n
LET K = \ f n . f^k n

iszero
prev

----

LET True = \ x y . x
LET False = \ x y . y

LET NOT = \ p . p False True

LET NEQ = \ x y . iszero (x prev y) (NOT (iszero (y prev x))) True

-}


-----------------------------------------------------
-- Název fldr a operátor +.+ použity, aby nodošlo ke kolizi se
-- standardní definicí


fldr _ a [] = a                             -- 1
fldr f a (x:xs) = f x (fldr f a xs)         -- 2

[]     +.+ xs = xs                          -- 3
(z:zs) +.+ xs = z:(zs +.+ xs)               -- 4


{-

forall ys
konečná xs

fldr (+) 0 xs + fldr (+) 0 ys = fldr (+) 0 (xs +.+ ys)

--------------------

1) xs = []
   fldr (+) 0 [] + fldr (+) 0 ys = fldr (+) 0 ([] +.+ ys)

L = fldr (+) 0 [] + fldr (+) 0 ys =|1
                0 + fldr (+) 0 ys =|aritmetika, přičtení nuly
                    fldr (+) 0 ys
P = fldr (+) 0 ([] +.+ ys) =|3
    fldr (+) 0 (ys) =|eliminace závorek
    fldr (+) 0 ys
L = P


2) xs = (a:as)
   fldr (+) 0 (a:as) + fldr (+) 0 ys = fldr (+) 0 ((a:as) +.+ ys)

I.P. fldr (+) 0 as + fldr (+) 0 ys = fldr (+) 0 (as +.+ ys)

L = fldr (+) 0 (a:as) + fldr (+) 0 ys =|2
    ((+) a (fldr (+) 0 as)) + fldr (+) 0 ys =|prefix->infix
    (a + (fldr (+) 0 as)) + fldr (+) 0 y =|asociativita +
    a + ((fldr (+) 0 as) + fldr (+) 0 y) =|I.P.
    a + fldr (+) 0 (as +.+ ys) =|infix->prefix
    (+) a (fldr (+) 0 (as +.+ ys)) =|2
    fldr (+) 0 (a:(as +.+ ys)) =|4
    fldr (+) 0 ((a:as) +.+ ys)

Q.E.D.

Rychlejší je fldr (+) 0 xs + fldr (+) 0 ys
-}

-----------------------------------------------------

data Name
  = Name String String Int
  deriving (Show, Eq)

data BTree a
  = BNode (BTree a) a (BTree a)
  | BLeaf
  deriving (Show, Eq)

readData :: FilePath -> IO (BTree Name)
readData fname = do
  h <- openFile fname ReadMode
  --
  c <- hGetContents h
  --
  res <- return $! toTree BLeaf (lines c)
  --
  hClose h
  --
  return res


toTree t [] = t
toTree t (l:ls) = toTree (ins2Tree t name) ls
  where
    (pr,r1) = span (/=':') l
    (jm,r2) = span (/=':') $ tail r1
    id = (read (tail r2))::Int
    name = Name pr jm id


ins2Tree BLeaf name = BNode BLeaf name BLeaf
ins2Tree (BNode l n r) nn =
  case cmpJm nn n of
    LT -> BNode (ins2Tree l nn) n r
    GT -> BNode l n (ins2Tree r nn)
    _  -> error "Duplicated items"


cmpJm (Name pr jm id) (Name p j i)
  | pr<p = LT
  | pr>p = GT
  | jm<j = LT
  | jm>j = GT
  | True = compare id i

-----------------------------------------------------
-- aleternativa z písemky, trochu upravené


data XName
  = XName String String Int
  deriving (Show, Eq, Ord)

data XTree a
  = XNode (XTree a) a (XTree a)
  | XLeaf
  deriving (Show, Eq)

readData' :: FilePath -> IO (XTree XName)
readData' fname = do
  h <- openFile fname ReadMode
  --
  c <- hGetContents h
  --
  res <- return $! toTree' XLeaf (lines c)
  --
  hClose h
  --
  return res


toTree' t [] = t
toTree' t (l:ls) = toTree' (ins2Tree' t name) ls
  where
    (pr,r1) = span (/=':') l
    (jm,r2) = span (/=':') $ tail r1
    id = (read (tail r2))::Int
    name = XName pr jm id


ins2Tree' XLeaf name = XNode XLeaf name XLeaf
ins2Tree' (XNode l n r) nn =
  if nn < n then XNode (ins2Tree' l nn) n r
            else XNode l n (ins2Tree' r nn)

-- rovnost netreba resit, v zadani bylo, ze jsou ruzne

-- EOF
```