```
import System.IO

-- 1

data LC a
  = Var a
  | App (LC a) (LC a)
  | Abs a (LC a)
  deriving (Show,Eq,Read)

betared :: Eq a => (LC a) -> (LC a) -> (LC a)
betared (Abs a body) ex = subst [] body
  where
    fvex =  fv ex
    subst bnds (Var v)
      | v/=a = Var v
      | foldr (||) False (map (\x -> x `elem` bnds) fvex) = error "Free variable occurs bound in the beta
reduction"
      | True = ex
    subst bnds (App e1 e2) = App (subst bnds e1) (subst bnds e2)
    subst bnds abs@(Abs x ex) = if x==a then abs else  Abs x (subst (x:bnds) ex)
betared _ _ = error "Just a lambda-abstraction can be applied"

fv (Var x) = [x]
fv (App e1 e2) = fv e1 ++ fv e2
fv (Abs v body) = filter (/=v) $ fv body


-- 2

lister l = map' f l     -- /1
  where
    f x = [x]           -- /2

map' f (x:xs) = f x : map' f xs  -- /3
map' _ _      = []               -- /4

{-

1)
Chci dokázat:
lister [] = []
====
     lister []
=|1  map' f []
=|4  []


2)
I.H.: lister as = ass

Chci dokázat:
lister (a:as) = [a] : ass
====
      lister (a:as)
=|1   map' f (a:as)
=|3   f a : map' f as
=|1   f a : lister as
=|IH  f a : ass
=|2   [a] : ass

Q.E.D.




2] variantní přístup
I.H.: lister as = [[a1], ... , [an]]

Chci dokázat:
lister (a:as) = [[a],[a1], ... , [an]]
===
      lister (a:as)
=|1   map' f (a:as)
=|3   f a : map' f as
=|1   f a : lister as
=|IH  f a : [[a1], ... , [an]]
=|2   [a] : [[a1], ... , [an]]
=|def(:) [[a],[a1], ... , [an]]

Q.E.D.

-- pozn: bylo možno uvést i definici (:) a [], ale to je zbytečné
-- pozn2: uznal jsem i definici s lister [] = [] i když potom byly všechny
   důkazy na něm založené špatně
-}


-- 3

fabc fi fo = do
```

```
  hi <- openFile fi ReadMode
  ho <- openFile fo WriteMode
  ct <- hGetContents hi
  let (res,val) = proc ct
  hPutStr ho (show val)
  if res then return () else hPutStr stderr "Error"
  hClose ho
  hClose hi

proc l
  | las==lbs && lbs==lcs && (length nocs==0) = (True,las)
  | True = (False,err)
  where
    (als,noas) = span (=='a') l
    las = length als
    --
    (bs,nobs) = span (=='b') noas
    lbs = length bs
    --
    (cs,nocs) = span (=='c') nobs
    lcs = length cs
    --
    err
      | las==0 = 1
      | las==lbs = if lcs>las then las+las+las+1 else las+las+lcs+1
      | True = if lbs>las then las+las+1 else las+lbs+1


-- span f l = (takeWhile f l, dropWhile f l)
-- pozor! takeWhile (=='a') a filter (=='a') není to samé!!!

-- EOF
```