```haskell
import IO

-- 1

{-

let T =  \ x y . x
let F =  \ x y . y

nechť Y je operátor pevného bodu, E je lambda-výraz
a k je pevný bod pro E, potom:

Y E = k = E k
Y E = E (Y E) ~ k = E k

LET minus = Y (\ f x y. iszero y x (f (prev x) (prev y)))

-}


-- 2

zp [] _ = []                          -- 1
zp _ [] = []                          -- 2
zp (x:xs) (y:ys) = (x,y) : zp xs ys   -- 3

zpW _ [] _ = []                               -- 4
zpW _ _ [] = []                               -- 5
zpW f (x:xs) (y:ys) = f x y : zpW f xs ys     -- 6

f x y = (x,y)  -- 7

{-

zp xs ys === zpW f xs ys

1)
  xs == [], forall ys!

  L = zp [] ys =|1 []
  P = zpW f [] ys =|4 []
  L = P

2)
  ys == [], forall xs!

  L = zp xs [] =|2 []
  P = zpW f xs [] =|5 []
  L = P

3)
  xs = (a:as), ys = (b:bs)
  I.P. = zp as bs = zpW f as bs

  L = zp (a:as) (b:bs) =|3
    = (a,b) : zp as bs =|IP
    = (a,b) : zpW f as bs =|7'
    = f a b : zpW f as bs =|6'
    = zpW f (a:as) (b:bs) = P

  Q.E.D.

-}


-- 3

insort [] = []
insort (x:xs) = foldr ins [x] xs
  where
     ins y [] = [y]
     ins y l@(z:zs) = if y > z then z : ins y zs else y : l


-- 4

data DLog
  = DVal Integer String
  | DNull
  deriving (Show,Eq)

notNullV :: DLog -> Bool
notNullV (DVal _ _) = True
notNullV _ = False

strV :: DLog -> String
strV (DVal i s) = show i ++ ":" ++ s
```

```haskell
pline :: String -> DLog
pline l =
  if null l then DNull else DVal ((read time)::Integer) val
  where
    time = takeWhile (\x -> elem x ['0'..'9']) l
    val = tail$ dropWhile (/='#') l

isM10 :: DLog -> Bool
isM10 (DVal i _) = (i `mod` 10) == 0

pt :: String -> IO ()
pt f = do
  h <- openFile f ReadMode
  c <- hGetContents h
  let ml = map pline $ lines c
  let nnl = filter notNullV ml
  let d10 = filter isM10 nnl
  putStrLn $ unlines $ map strV d10
  hClose h


-- Prémie
-- pro test je možné využít toto prvočíslo 5000000029

isPrv = tst prv
  where
    prv = 2 : filter (tst prv) [3,5 ..]
    tst (p:ps) x = (p*p)>x || ((x `mod` p /=0) && tst ps x)


-- EOF
```