1 Problem

The decimal number, $585 = 1001001001_2$ (binary), is palindromic in both bases.

Find the sum of all numbers, less than one million, which are palindromic in base 10 and base 2.

(Please note that the palindromic number, in either base, may not include leading zeros.)

2 Solution

```
import Data.List
import qualified Data.Map as Map
import Data.Maybe
import System.Environment

data BaseNumber = BaseNumber
  { digits :: [Int]
  , nBase :: Int
  } deriving (Read)

instance Show BaseNumber where
  show bn = dstring ++ "_[" ++ bstring ++ "]"
    where dstring = filter (\z -> z `notElem` ",[]") (show $ digits bn)
          bstring = show $ nBase bn

baseToDecimal :: BaseNumber -> Int
baseToDecimal bn
  | digits bn == [] =  0 :: Int
  | otherwise = dig + (nBase bn) * (baseToDecimal bn')
    where dig = last $ digits bn
          bn' = BaseNumber (init $ digits bn) (nBase bn)

decimalToBase :: Int -> Int -> BaseNumber
decimalToBase dec bs = BaseNumber (getBaseDigit dec bs ord) bs
  where ord = (floor $ logBase (fromIntegral bs) (fromIntegral dec))

getBaseDigit :: Int -> Int -> Int -> [Int]
getBaseDigit dec _ 0 = [dec]
getBaseDigit dec bs ord = concat [[d], getBaseDigit dec' bs ord']
  where ord' = ord - 1
        d = dec `div` (bs ^ ord)
        dec' = dec - (d * bs ^ ord)

class BNumber a where
  isPalindrome :: a -> Bool

instance BNumber Int where
  isPalindrome a = ((show a) == ((reverse . show) a))

instance BNumber BaseNumber where
  isPalindrome a = ((digits a) == ((reverse . digits) a))

main = do
  let allDoublePals = filter (\z -> (isPalindrome z) &&
        (isPalindrome (decimalToBase z 2))) ([1..1000000] :: [Int])
      sumAll = sum allDoublePals
  putStrLn $ "The sum of all numbers less than 1000000 which are " ++
    "palindromes in both binary and decimal is " ++ show sumAll ++ "."
```

3 Result

```
runhaskell problem36.lhs
The sum of all numbers less than 1000000 which are palindromes
in both binary and decimal is 872187.
```