1 Problem

If we take 47, reverse and add, $47 + 74 = 121$, which is palindromic.

Not all numbers produce palindromes so quickly. For example,

$$349 + 943 = 12921292 + 2921 = 42134213 + 3124 = 7337$$

That is, 349 took three iterations to arrive at a palindrome.

Although no one has proved it yet, it is thought that some numbers, like 196, never produce a palindrome. A number that never forms a palindrome through the reverse and add process is called a Lychrel number. Due to the theoretical nature of these numbers, and for the purpose of this problem, we shall assume that a number is Lychrel until proven otherwise. In addition you are given that for every number below ten-thousand, it will either

1. become a palindrome in less than fifty iterations, or,
2. no one, with all the computing power that exists, has managed so far to map it to a palindrome.

In fact, 10677 is the first number to be shown to require over fifty iterations before producing a palindrome: 4668731596684224866951378664 (53 iterations, 28-digits).

Surprisingly, there are palindromic numbers that are themselves Lychrel numbers; the first example is 4994.

How many Lychrel numbers are there below ten-thousand?

NOTE: Wording was modified slightly on 24 April 2007 to emphasise the theoretical nature of Lychrel numbers.

2 Solution

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

isPalindrome x = (show x) == (reverse $ show x)

isLychrel x = isLychrel' 50 x

isLychrel' 0 x = True
isLychrel' n x =
  let x' = x + (read (reverse $ show x) :: Integer)
  in if isPalindrome x'
    then False
    else isLychrel' (n - 1) x'

main = do
  let soln = length $ filter isLychrel [1..9999]
  putStrLn $ "There are " ++ show soln ++ " Lychrel numbers below " ++
    "ten thousand."
```

3 Result

```
runhaskell problem55.lhs
There are 249 Lychrel numbers below ten thousand.
```