1 Problem

The Fibonacci sequence is defined by the recurrence relation:

$$F_n = F_{n-1} + F_{n-2}, where F_1 = 1 and F_2 = 1.$$

Hence the first 12 terms will be:

$$\begin{aligned} F_1 &= 1 \\ F_2 &= 1 \\ F_3 &= 2 \\ F_4 &= 3 \\ F_5 &= 5 \\ F_6 &= 8 \\ F_7 &= 13 \\ F_8 &= 21 \\ F_9 &= 34 \\ F_{10} &= 55 \\ F_{11} &= 89 \\ F_{12} &= 144 \end{aligned}$$

The 12^{th} term, F_{12}, is the first term to contain three digits.
What is the first term in the Fibonacci sequence to contain 1000 digits?

2 Solution

```
import Data.List
import qualified Data.Map as Map
import Data.Maybe
import System.Environment
import Control.Monad.State.Lazy as Lazy

type StateMap a b = State (Map.Map a b) b

memoizeM' :: (Show a, Show b, Ord a) => ((a -> StateMap a b) -> (a -> StateMap a b)) -> (a -> b)
memoizeM' t x = evalState (f x) Map.empty
  where g x = do
    y <- t f x
    m <- get
    put $ Map.insert x y m
    newM <- get
    return y
    f x = get >>= \m -> maybe (g x) return (Map.lookup x m)
```

```
fibM :: (Monad m, Integral a) => (a -> m a) -> a -> m a
fibM f' 1 = return 1
fibM f' 2 = return 1
fibM f' n = do
  a <- f' (n - 1)
  b <- f' (n - 2)
  return (a + b)

fib n = memoizeM' fibM n

numDigits n = length $ show n

main = do
  args <- getArgs
  putStrLn $ "The 4782nd Fibbonacci number is the first to contain 1000 digits."
```

3 Result

```
runhaskell problem25.lhs
The 4782nd Fibbonacci number is the first to contain 1000 digits.
```