1 Problem

The following iterative sequence is defined for the set of positive integers:

$$n \to n/2 (niseven) \tag{1}$$
$$n \to 3n + 1 (nisodd) \tag{2}$$
$$\tag{3}$$

Using the rule above and starting with 13, we generate the following sequence:

$$13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$$

It can be seen that this sequence (starting at 13 and finishing at 1) contains 10 terms. Although it has not been proved yet (Collatz Problem), it is thought that all starting numbers finish at 1.

Which starting number, under one million, produces the longest chain?

NOTE: Once the chain starts the terms are allowed to go above one million.

2 Solution

```
import System.Environment
import Debug.Trace
import Data.Map as Map
import Control.Monad.State.Lazy as State

type StateMap a b = State (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 $ trace ("Map now contains" ++ Map.showTree newM) y
  f x = get >>= λm → maybe (g x) return (Map.lookup x m)
twond m n = trace ("wondM called with " ++ show m ++ " returning " ++ show n) n
wondM :: Monad m ⇒ (Integer → m Integer) → Integer → m Integer
wondM f' 1 = return $ twond 1 1
wondM f' n = do
  let n' = if even n
        then (n `div` 2)
        else (3 * n + 1)
```

```
  n'' ← f' n'
  return $ twond n (1 + n'')
wond n = memoizeM wondM n
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 >>= λm → maybe (g x) return (Map.lookup x m)
wondM' :: Monad m ⇒ (Integer → m Integer) → Integer → m Integer
wondM' f' 1 = return 1
wondM' f' n = do
  let n' = if even n
           then (n `div` 2)
           else (3 * n + 1)
  n'' ← f' n'
  return (1 + n'')
wond' n = memoizeM' wondM' n
main = do
  args ← getArgs
  let n = read (args !! 0) :: Integer
  let maxOver = maximum $ Prelude.map wond' [1 .. n]
  let maxWith = Prelude.filter (λz → wond' z ≡ maxOver) [1 .. n]
  putStrLn $ "The maximum value achieved is " ++ show maxOver ++ " which is found using n =
```