Code (Wednesday Week 7)

Leaky Hash Function

Haskell

```haskell
module Hash where
import Data.Hashable
import Test.QuickCheck
import Test.QuickCheck.Monadic
import Test.QuickCheck.Modifiers

newtype Password = Password String
                   deriving (Show, Eq)

instance Arbitrary Password where
  arbitrary = fmap Password $ vectorOf 8
            $ elements (['A'..'Z']++['a'..'z']++['0'..'9'])

hashIO :: String -> IO Int
hashIO x = do
             appendFile "leaked-info" (x ++ "\n")
             pure (hash x)

prop_hash_same :: Password -> Property
prop_hash_same (Password s) = monadicIO $ do
                    h <- run (hashIO s)
                    assert (h == hash s)

-- purity in the absence of a pure model
prop_hash_independent :: Password -> Property
prop_hash_independent (Password s) = monadicIO $ do
                    h <- run (hashIO s)
                    h' <- run (hashIO s)
                    assert (h == h')
```

Stateful Fibonacci

```haskell
import Control.Monad.State.Strict
import Test.QuickCheck
import Test.QuickCheck.Modifiers

-- >>
seqComp :: Monad m => m a -> m b -> m b
```

```haskell
seqComp a b = a >>= \_ -> b

done :: Applicative f => f ()
done = pure ()

repeatFor :: Monad m => Int -> m () -> m ()
repeatFor n action = foldr seqComp done (replicate n action)

fib :: Int -> Integer
fib 0 = 0
fib 1 = 1
fib n = fib (n-1) + fib (n-2)

fib2 :: Int -> Integer
fib2 n = fst (fib' n)
            where fib' 0 = (0,1)
                  fib' n = (b, a+b) where (a,b) = fib' (n-1)

fibState' :: Int -> State (Integer, Integer) Integer
fibState' n = do
              repeatFor n
                (do
                  (a,b) <- get
                  put (b, a+b))
              (x,y) <- get
              pure x

fibState :: Int -> Integer
fibState n = evalState (fibState' n) (0,1)

prop_fibState_same_tiny :: Property
prop_fibState_same_tiny
   = forAll (elements [0..30])
       (\n -> fibState n == fib n)

prop_fibState_same :: Property
prop_fibState_same
   = forAll (elements [0..10000])
       (\n -> fibState n == fib2 n)

-- performance matters when we are testing our abstract model
```