Code (Wednesday Week 9)

Curry-Howard Correspondence

Haskell

```haskell
-- a -> a is a type but also a proposition
prop :: a -> a
-- \a -> a is a program but it is also a constructive
-- proof of the proposition
--    assume a, we have a
prop a = a

-- prop2 :: (a -> b) -> b

prop2 :: a -> (Either a void)
-- a -> (a \/ false)
prop2 a = Left a

prop3 :: a -> Either a ()
-- a -> (a \/ true)
prop3 a = Right ()

prop4 :: a -> (a,())
-- a -> a /\ true
prop4 a = (a,())

prop_semi_fancy:: a -> (a-> void) -> void
-- a -> (a-> false) -> false
-- ~ a = (a-> false)
-- a -> ~ a -> false
prop_semi_fancy a f = f a


prop_fancy :: (((a -> void) -> void) -> void) -> a -> void
-- (((a -> false) -> false) -> false) -> a -> false
-- (~a -> false) -> false) -> a -> false
-- (~ (~a)) -> false) -> a -> false
-- ~ (~ (~a)) -> a -> false
-- ~ (~ (~a)) -> ~a
prop_fancy f a = f (\g -> g a)
```