Chapter 14

Exercise 1: Complete the following instance declaration from `Data.Monoid` to make a pair type into a monoid provided the two component types are monoids:

```
data Pair a b = Pair a b

instance (Semigroup a, Semigroup b) => Semigroup (Pair a b) where
    (<>) :: Pair a b -> Pair a b -> Pair a b
    Pair x1 y1 <> Pair x2 y2 = Pair (x1 <> x2) (y1 <> y2)

instance (Monoid a, Monoid b) => Monoid (Pair a b) where
    mempty :: Pair a b
    mempty = Pair mempty mempty
```

Exercise 2: In a similar manner, show how a function `type a -> b` can be made into a monoid provided that the result type `b` is a monoid.

```
newtype Fun a b = Fun (a -> b)

instance (Semigroup b) => Semigroup (Fun a b) where
    (<>) :: Fun a b -> Fun a b -> Fun a b
    Fun f <> Fun g = Fun $ \x -> f x <> g x

instance (Monoid b) => Monoid (Fun a b) where
    mempty :: Fun a b
    mempty = Fun $ const mempty
```