Functors

Write `Functor` instances for the following three data types (don't forget to look at the **hint** at the end of this exercise!)

- Write a `Functor` instance for `Identity` which just wraps a value.

  ```
  data Identity a = Identity a deriving Show
  ```

  Example

  ```
  > fmap not (Identity False)
  Identity True
  ```

- Write a `Functor` instance for `Pair a` which combines two types. Note that `fmap` only looks at the second element of the pair.

  ```
  data Pair a b = Pair a b deriving Show
  ```

  Example

  ```
  > fmap length (Pair 'z' "zet")
  Pair 'z' 3
  ```

- Write a `Functor` instance for `Unit` which contains no value, so its type parameter can be chosen freely, while the same constructor can be used.

  ```
  data Unit a = Unit deriving Show
  ```

  Example

  ```
  > fmap (++ [True]) Unit
  Unit
  > fmap not Unit
  Unit
  ```

Hint Recall that a `Functor` instance has the following form (where everything between `<...>` must be replaced):

```
instance Functor <type> where
    fmap f (<constructor> <0 or more arguments>) = <result>
```

For example:

```
instance Functor Maybe where
    fmap _ Nothing  = Nothing
    fmap f (Just x) = Just (f x)
```