Exercises on Monads 19/12/2016

1) Define an instance of the Monad class for the type (a ->) . Remember that one has to write, instance Monad ((->)a) where...

2) Given the following type of expressions

```
data Expr a = Var a | Val Int | Add (Expr a) (Expr a) deriving Show
```

that contains variables of some tipe a, show how to make this type into instances of Functor, Applicative and Monad classes. With the aid of an example, explain what the >>= operator for this type does.

3) Rather than making a parameterized type into instances of the Funcor, Applicative and Monad classes in this order, in practice it is sometimes simpler to define the Functor and Applicative instances in terms of the Monad instance, relying on the fact that the order in which the declarations are made is not important in Haskell. Thus, given

```
instance Monad ST where
st >>= f = S(\s ->
        let (x,s') = app st s in app (f x) s')
```

define the Functor and Applicative instances of ST, using the do notation.