Question 2

```
type State = Int
newtype ST a = S (State -> (a,State))

apply :: ST a -> State -> (a, State)
apply (S st) x = st x

instance Functor ST where
  -- fmap :: (a -> b) -> ST a -> ST b
  fmap g st = st >>= (pure . g)

instance Applicative ST where
  -- pure :: a -> ST a
  pure x = S (\s -> (x,s))

  --(<*>) :: ST (a -> b) -> ST a -> ST b
  stf <*> stx = stf >>= (\f -> fmap f stx)

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