exercises
State Monads

Exercise 1:

Give the example of a stack in slide deck, write a similar program to manage a Queue structure. A Queue can be seen as a List of elements. The main functions of a Queue are

1. ***enQueue*** (an element is added to the end of the list - similar to ***push*** in stack)
2. ***deQueue*** (an element is returned from the top of the list - similar to ***pop*** in stack)

Write a short program to test the operation of the queue (similar to the example for the stack)

Solution 1:

```
type Queue a = [a]

-- Add an element to the end of the queue
enQueue :: a -> Queue a -> Queue a
enQueue x q = q ++ [x]

-- Remove an element from the front of the queue
deQueue :: Queue a -> (Maybe a, Queue a)
deQueue []     = (Nothing, [])           -- Empty queue case
deQueue (x:xs) = (Just x, xs)

-- Function to display the queue
printQueue :: Show a => Queue a -> IO ()
printQueue q = putStrLn $ "Queue: " ++ show q

-- Test program
main :: IO ()
main = do
        -- Start with an empty queue
        let q0 = [] :: Queue Int
        printQueue q0

        -- Enqueue some elements
        let q1 = enQueue 10 q0
        let q2 = enQueue 20 q1
        let q3 = enQueue 30 q2
        printQueue q3

        -- Dequeue an element
        let (e1, q4) = deQueue q3
        putStrLn $ "Dequeued: " ++ show e1
        printQueue q4

        -- Dequeue another element
        let (e2, q5) = deQueue q4
        putStrLn $ "Dequeued: " ++ show e2
        printQueue q5
```