CSE 341 — Haskell Mini-Exercises # 2

These are questions for discussion in class. (You don't need to hand in anything.) The solutions are on the class web page.

1. Write a pointfree function `rev_square` that takes a list of integers and returns their squares, in reverse order.
2. Write a function `concat'` that concatenates a list of lists. Use `foldr`. (There is a function `concat` in the Prelude that does this, hence the different name.)
3. Suppose that we have the following definition of the `member` function in Haskell:

   ```
   member x [] = False
   member x (y:ys) | x==y      = True
                   | otherwise = member x ys
   ```

   Circle each type declaration that is a correct type for `member`. (Not necessarily the most general type, just a correct one.)

   ```
   member :: Integer -> Integer -> Bool

   member :: (Ord a) => a -> [a] -> Bool

   member :: (Integer -> Integer) ->  [Integer -> Integer] -> Bool

   member :: (Eq a) => a -> [a] -> Bool

   member :: a -> [a] -> Bool

   member :: (Eq a) => [a] -> [[a]] -> Bool

   member :: Bool -> [Bool] -> Bool
   ```

   Which of the above types, if any, is the most general type for `member`?
4. The `TypesNotes.hs` lecture notes include a `preorder` function that does a pre-order traversal on the newly defined `Tree` datatype. Define `inorder` and `postorder` functions as well.
5. Write a Haskell type `List` that is like built-in lists, but defined from scratch.
6. Write a Haskell function `append` that works on instances of the `List` type. What is the type of this function?
7. Write a Haskell function `mymap`, like the built in map but that works on instances of the `List` type. What is the type of this function?

8. Write a Haskell action `capitalize` that reads in a line of text and prints it out in all capitals. (Hint: use the function `Data.Char.toUpper`.)

9. Write a Haskell action `santa` that takes a parameter `n`, and prints out `ho` that many times. What is the type of `santa`?

10. Convert the following actions into equivalent ones that don't use `do`:

```
printsqrt2 = do
  putStr "the square root of 2 is "
  putStrLn (show (sqrt 2))

calcsqrt = do
  x <- readLn
  putStrLn "calculating the square root of x"
  putStrLn (show (sqrt x))
```