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 \( \text{rev\_square} \) that takes a list of integers and returns their squares, in reverse order.

2. Write a function \( \text{concat'} \) that concatenates a list of lists. Use \( \text{foldr} \). (There is a function \( \text{concat} \) in the Prelude that does this, hence the different name.)

3. Suppose that we have the following definition of the \( \text{member} \) function in Haskell:

\[
\begin{align*}
\text{member} \; x \; [] & = \text{False} \\
\text{member} \; x \; (y:ys) & | x == y = \text{True} \\
& | \text{otherwise} = \text{member} \; x \; ys
\end{align*}
\]

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

\[
\begin{align*}
\text{member} &: \text{Integer} \to \text{Integer} \to \text{Bool} \\
\text{member} &: (\text{Ord} \; a) \Rightarrow a \to [a] \to \text{Bool} \\
\text{member} &: (\text{Integer} \to \text{Integer}) \to [\text{Integer} \to \text{Integer}] \to \text{Bool} \\
\text{member} &: (\text{Eq} \; a) \Rightarrow a \to [a] \to \text{Bool} \\
\text{member} &: a \to [a] \to \text{Bool} \\
\text{member} &: (\text{Eq} \; a) \Rightarrow [a] \to [[[a]]] \to \text{Bool} \\
\text{member} &: \text{Bool} \to [\text{Bool}] \to \text{Bool}
\end{align*}
\]

Which of the above types, if any, is the most general type for \( \text{member} \)?

4. The \texttt{TypesNotes.hs} lecture notes include a \texttt{preorder} function that does a pre-order traversal on the newly defined \texttt{Tree} datatype. Define \texttt{inorder} and \texttt{postorder} functions as well.

5. Write a Haskell type \texttt{List} that is like built-in lists, but defined from scratch.

6. Write a Haskell function \texttt{append} that works on instances of the \texttt{List} type. What is the type of this function?

7. Write a Haskell function \texttt{mymap}, like the built in in map but that works on instances of the \texttt{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:

```hs
printsqrt2 = do
    putStrLn "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))
```