1. Suppose that we have the following definition of the `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 `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`?

2. What are the first ten elements in the following list?

\[
mystery = 1 : \text{map } (*2) \; \text{mystery}
\]

3. Define a variable `ints` whose value is the infinite list of all integers. It should be ordered in such a way that you can find any given integer after searching a finite number of elements in `ints`. In other words, this isn’t going to work:

\[
\text{ints} = [1 ..] ++ [-1, -2 ..]
\]

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

\[
\text{printsqrt2} = \text{do} \\
\text{putStr } "\text{the square root of 2 is } " \\
\text{putStrLn } (\text{show } (\text{sqrt 2}))
\]

\[
\text{calcsqrt} = \text{do} \\
\text{x } <- \text{readLn} \\
\text{putStrLn } "\text{calculating the square root of } x" \\
\text{putStrLn } (\text{show } (\text{sqrt } x))
\]