COMP 3649 Exercise Set 5

Given: Monday, March 3
Complete by: Monday, March 10
Quiz: Tuesday, March 11 (in-class at 8:30 a.m.)

For questions 1-4, attempt to develop written answers by yourself. Once you have done your best, you may discuss your answers with two-to-three other students. Then, by yourself (without looking at notes from the discussion), revise your own written answers as you believe appropriate.

1. Write a function that takes a list of years (as integers) and keeps only those that are leap years. Define the function by calling `filter`. You may define an `isLeapYear` helper function as:

```
isLeapYear y = divsBy 4 && (not (divsBy 100) || divsBy 400)
    where divsBy d = y `mod` d == 0
```

2. Write a function that takes a list of years (as integers) and determines if any are leap years. Do this two ways.

    a) First, by using the function from the previous question
    b) Second, by mapping the list of integers to a list of Booleans and then using "big or"

    Finally, comment on whether one of (a) or (b) is more efficient than the other version. Explain your answer.

3. Define a higher-order `takewhile` function that behaves as follows[1]:

```
ghci> takewhile (<5) [1,2,1,4,5,2,3]
[1,2,1,4]
ghci> takewhile isLetter "hello 2 u!"
"hello"
```

4. Assuming the existence of the `fibs` function from lecture that produced the infinite list of Fibonacci numbers, the following list comprehension can "hang".

```
-- supposed to give the finite list of cubes of even fibs less
-- than 20:

[ x^3 | x <- fibs, even x, x < 20 ]
```

    Show how to correct it using `takewhile`.

[1] `isLetter` is defined in `Data.Char`.

5. Show an example of a list comprehension (not already used as an example in class) that includes both filtering and mapping aspects.

   Next, show an equivalent expression that evaluates to the same result (i.e., that produces the same list) without using a list comprehension but instead making explicit use of the higher-order `map` and `filter` functions.

Develop complete answers to questions 6-8 working in a group of two-to-four students (but ensure you can recreate a correct answer on your own).

6. Define a higher-order `fold` function generalized from definitions such as the following:

```
sum    []     = 0
sum    (x:xs) = x +  sum    xs

and    []     = True
and    (x:xs) = x && and    xs

length []     = 0
length (x:xs) = 1 +  length xs
```

   For example, it should be possible to redefine `sum` and `and` as follows:

```
sum xs = fold (+) 0 xs
and xs = fold (&&) True xs
```

   You may not use the standard `foldl` or `foldr` functions as part of your answer.

   Hint: the type should be `(a -> b -> b) -> b -> [a] -> b`

7. Show how to redefine `length` in one line, using `fold` from the previous question.

8. Consider a polymorphic binary tree data type:

```
data BinTree a = Leaf | Node a (BinTree a) (BinTree a)
```

   Define a polymorphic, higher-order function for mapping trees.