COMP 3649 Exercise Set 2

Given: Tuesday, January 28
Complete by: Monday, February 3
Quiz: Tuesday, February 4 (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 (declare and define) a function that computes the area of a circle given its radius as a `Double`.

2. Consider the data type definition below (equivalent to the pre-defined version in the standard Prelude):

```
data Bool = True | False
```

   Define your own versions of two-argument "logical or" function (pre-defined in the Prelude as `(||)` but not to be used in this exercise) *without* invoking any other functions, in the following ways:

   a) First, without using any variables (or "don't cares")[1];
   b) Second, a shorter way, using variables (possibly including "don't cares");
   c) Third, in one line using the pre-defined `not` and (`&&`) operators.

   Your definitions must include type declarations.

3. You should always know the type of a function before you attempt to define it.

   Based on an informal description, what should be the type of each of the following functions? Write a declaration (not a definition) using correct Haskell syntax.

   You are to assume the existence of the following types: `Int`, `Char`, `Bool`.

   a) `majorityAboveAvg` takes three integers and determines if most are above average, or not.

   b) `toLower` coverts its input to lower case or leaves the input unchanged when it isn't a capital letter.

   c) `retirementAge` – a constant.

4. Write (declare and define) a function that determines if a given list is empty.

[1] Hint: think of the truth table for the operation, and then construct a four-case definition using pattern matching.

Develop complete answers to questions 5-6 working in a group of two-to-four students (but ensure you can recreate a correct answer on your own).

5. The standard prelude contains an `and` function that takes a list of Booleans and returns `True` if and only if none are `False`.

   E.g.

   ```
   and [True,True,True]          = True
   and [True,False,True]         = False
   and []                        = True
   ```

   Write this function yourself.

   Note: if you wish to test your work in the interpreter and wish to name your function `and`, place the following at the top of your script:

   ```
   import Prelude hiding (and)
   ```

   Otherwise, your definition will clash with the standard prelude's definition (recall that the prelude is, by default, loaded first). An error will result.

6. Given your function definition from the previous question, hand-evaluate the following two expressions. Follow the format presented in lecture.

   a) `and [True,True,True]`
   b) `and [True,False,True,True]`

   Assume and use the following definition in your work:

   ```
   False && _ = False          -- (&&).1
   True  && b = b              -- (&&).2
   ```

For question 7, attempt to develop a written answer by yourself. Once you have done your best, you may discuss your answer 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.

7. The standard prelude contains an "index" operator (`!!`) for lists. It returns the element at the given index in the given list.

   E.g.,

   ```
   ['a','b','c'] !! 0      = 'a'
   "abc"         !! 2      = 'c'
   ```

   Write this function yourself.

Note:

- The second argument must be an `Int`. Otherwise, the type must be as general as possible.
- You may omit the handling of cases that are out-of-bounds. They should simply result in an evaluation-time error.

Again, to test your definition in the interpreter, if using the name `(!!)`, you will need something like the following:

```
import Prelude hiding (and,(!!))
```

For questions 8-10, 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.

8. The standard prelude includes a `replicate` function:

   ```
   replicate :: Int -> a -> [a]
   ```

   This function produces a list with the given number of copies of the given value.

   ```
   replicate 3 1                 = [1,1,1]
   replicate 2 [True,False]      = [[True,False],[True,False]]
   replicate (-1) 3.7            = []
   ```

   Write `replicate` yourself.

9. Think of an example, not already discussed in class but potentially implemented in an exercise above, of each of the following categories of functions overs lists:

   - fold
   - map
   - filter

   Write (declare and define) each. If you select as example one of the functions you've already defined in a previous exercise, then there is no need to redefine it here.

10. Implement `majorityAboveAvg` from a previous exercise, above.