Haskell Review 1

1. Write a function `radius :: Double -> Double -> Double` which takes two floating-point numbers, x and y, and returns the distance from the point (x, y) to the origin.

2. Write a function `radius' :: (Double,Double) -> Double` which operates the same way as above, but it takes as input the *pair* (x, y) as a single argument.

3. Write a function `sumEvens :: Integer -> Integer` which adds up all the even numbers from 1 to its input argument.
   ```
   sumEvens 5 = 6
   sumEvens 8 = 20
   ```
   Do not use list comprehension or Haskell's built-in "sum" function.

4. Write a function `sumEvens' :: Integer -> Integer` which does the same thing as the previous one, but now using list comprehension and other functions.

5. The *Collatz function* is defined as following:

$$f(n) = \begin{cases} 1 & n = 0 \text{ or } n = 1 \\ f(n/2) & n > 1 \text{ is even} \\ f(3n+1) & n > 1 \text{ is odd} \end{cases}$$

   Write a Haskell function `collatz :: Integer -> Integer` so that `collatz n` equals $f(n)$ for every $n \geq 0$.

6. The *Collatz Conjecture* states that $f(n)$ always returns 1 when $n > 0$. No one knows if this is true. You will be very famous if you solve this problem, see `https://en.wikipedia.org/wiki/Collatz_conjecture`.

   Verify that the Collatz conjecture is true for $1 \leq n \leq 100$, by constructing a list `collatzCheck :: [Integer]` consisting of the values of $f(n)$ in that range.

7. Using ranges and/or list comprehension, create a list of all numbers $1 \leq n \leq 100$ divisible by 5. Save this as a Haskell term `multiplesOfFive :: [Integer]`.

8. Using Haskell's built-in functions, write a function `init'` which returns all but the last element. Do *not* use Haskell's built-in `init` function.

9. Using Haskell's built-in functions, write a function `findEmpty :: [String] -> Bool` that takes a list of strings, and determines whether it contains an empty string. (*Hint.* Remember that strings are really lists, so you can use functions like `elem` and `null`.)

10. Using Haskell's built-in functions and/or list comprehension, write a function `getLengths :: [String] -> [Int]` which takes a list of strings and returns a list containing their *lengths*:
    ```
    getLengths ["Hello","World"] = [5,5]
    ```

Submission instructions

- Save your file as `LASTNAMErev1.hs`. Submit it to asulearn by the deadline.
- *Your file must compile/load correctly.*

  If your file does not load, gives a type error or some formatting error, you will not receive any credit for this assignment.
- *You should provide a definition for every function in the assignment.*

  If you can't get a function to work, use the `error` construct to complete the answer to that problem. For example,

```
collatz :: Integer -> Integer
collatz n = error "not done"
```