Lab 8 Continued:

3) Validating Credit Card Numbers

Have you ever wondered how websites validate your credit card number when you shop online? They don't check a massive database of numbers, and they don't use magic. In fact, most credit providers rely on a checksum formula called the Luhn Algorithm for distinguishing valid numbers from random collections of digits (or typing mistakes).

In this assignment, you will implement the Luhn Algorithm. Pseudocode for the algorithm is provided below:

- Double the value of every second digit beginning from the right. That is, the last digit is unchanged; the second-to-last digit is doubled; the third-to-last digit is unchanged; and so on. For example, [5,5,9,4] becomes [10,5,18,4].

- Add the digits of the doubled values and the undoubled digits from the original number. For example, [10,5,18,4] becomes (1 + 0) + 5 + (1 + 8) + 4 = 19

- Calculate the remainder when the sum is divided by 10. For the above example, the remainder would be 9.

- If the result equals 0, then the number is valid.

The following series of exercises will lead you to a completed implementation of the algorithm. In your code, clearly mark each step with Haskell comments.

Exercise 1: We first need to be able to break up a number into its last digit and the rest of the number. Write these functions:

lastDigit :: Integer -> Integer
dropLastDigit :: Integer -> Integer

- Example: **lastDigit 123 == 3**
- Example: **lastDigit 0 == 0**
- Example: **dropLastDigit 123 == 12**
- Example: **dropLastDigit 5 == 0**

Note that some test cases for these functions have been provided in LAB8Tests.hs. To run the tests for this exercise, load LAB8Tests.hs in to GHCi and type **runTests ex1Tests**. The result is a list of failures; if the list is empty then all of the tests passed. You should add your own test cases for the remaining exercises.

Exercise 2: Now, we can break apart a number into its digits. It is actually easier to break a number in to a list of its digits in reverse order (can you figure out why?). Your task is to define the function

toRevDigits :: Integer -> [Integer]
toRevDigits should convert positive **Integer**s to a list of digits. (For **0** or negative inputs, **toRevDigits** should return the empty list.)

- Example: **toRevDigits 1234 == [4,3,2,1]**
- Example: **toRevDigits 0 == []**
- Example: **toRevDigits (-17) == []**

It is easy to define a function that gets a list of digits in the proper order in terms of **toRevDigits**:

toDigits :: Integer -> [Integer]
toDigits n = reverse (toRevDigits n)

However, you will likely not need to use it for this lab (you're welcome to implement it from scratch as an auxiliary exercise).

Exercise 3: Once we have the digits in a list, we need to double every other one. Define a function

doubleEveryOther :: [Integer] -> [Integer]

Remember that the **Luhn** algorithm should double every other digit beginning from the right, that is, the second-to-last, fourth-to-last, . . . numbers are doubled. It's much easier to perform this operation on a list of digits that's in reverse order. Conveniently, the function you defined in the previous exercise gives you the digits in reverse order. The function **doubleEveryOther** should thus take in a list that is already in reverse order and double every other number starting with the second one.

- Example: **doubleEveryOther [4, 9, 5, 5]** = **[4, 18, 5, 10]**
- Example: **doubleEveryOther [0, 0]** = **[0, 0]**

Exercise 4: The output of **doubleEveryOther** has a mix of one-digit and two-digit numbers. Define the function sumDigits :: [Integer] -> Integer
to calculate the sum of all digits.

- Example: sumDigits [10, 5, 18, 4] = 1 + 0 + 5 + 1 + 8 + 4 = 19

Exercise 5: Define the function
luhn :: Integer -> Bool

that indicates whether an **Integer** could be a valid credit card number. This should use all functions defined in the previous exercises. Remember that your implementation of **doubleEveryOther** gives you a list in reverse order. Do you need to account for this?

- Example: **luhn** 5594589764218858 = **True**
- Example: **luhn** 1234567898765432 = **False**

If you want more credit card numbers to test on you can get some at http: //www.getcreditcardnumbers.com.

You can test on your own credit cards as well, but you probably don't want to submit test cases with your credit card information.

Submission Upload your warmup.hs, LAB8.hs (with function definitions) and LAB8Tests.hs .