HOMEWORK 2: CS 4450

DUE: 3:00PM, 10/16/17

Directions

* Name your submission file as follows. For groups, submission files must be named `{groupname}_homework2.hs`. For example, a group named `reynolds` would submit `reynolds_homework2.hs`. For individuals, submission files must be named `{pawprint}_homework2.hs`. For example, an individual with a pawprint `tnrn9b` would submit `tnrn9b_homework2.hs`.
* Your submission *must* load and typecheck in the Haskell Platform to get any points.
* Every function you define must have a type signature in order to get full credit for the problem.
* Functions must be documented. Examples provided in the assignment file.
* Name all functions and data types exactly as they appear in the assignment. You cannot import any additional libraries or enable any additional language features unless granted explicit permission from the instructor.
* The code you submit must be your own. *Exceptions*: you may (of course) use the code we provide however you like, including examples from the slides.
* NO LATE SUBMISSIONS! Don't submit the assignment hours after it is due and claim that Canvas wasn't working. If Canvas truly isn't working, you should submit the assignment via email minutes after it is due (3:00 PM) – NOT HOURS AFTER IT IS DUE.
* Each problem has at least one test written using the `QuickCheck` package. Make sure you have `QuickCheck` installed! In GHCI, use `test_prob1`, `test_prob2`, `test_prob3`, `test_prob4`, or `test_prob5` to test individual problems, or `test_probs` to test all problems.

Problems

1. Define a function, `prob1`, by rewriting `listComp` using the functions `map` and `filter` instead of list comprehension:

   ```
   listComp f p xs = [ f x | x <- xs, p x ]
   ```

   Keep in mind that the inputs are `f`, a function, `p` a function the returns a `Bool`, and `xs`, a list of appropriate type. Failure to use both `map` and `filter` will result in 0 points for this problem.
2. Define a function `prob2` that takes an `Integer` as input and returns a list of digits, i.e., `[Integer]`, in order as they occur in the input. For negative inputs, it should return `[]`.

   ```
   *Homework2> prob2 (-1)
   []
   *Homework2> prob2 967896
   [9,6,7,8,9,6]
   ```
3. Define a function `prob3` that takes an `Integer` as input and returns a list of digits, i.e., `[Integer]` in REVERSE order as they occur in the input. For negative inputs, it should return `[]`.

   ```
   *Homework2> prob3 (-1)
   []
   *Homework2> prob3 967896
   [6,9,8,7,6,9]
   ```
4. Define a function `prob4` that takes a list of non-negative numbers, i.e., `[Integer]`, and multiplies every other digit starting from the *right* by 2.

   ```
   *Homework2> prob4 [18,1,2,18]
   [36,1,4,18]
   *Homework2> prob4 []
   []
   ```
5. Define a function `prob5` that takes a list of non-negative numbers, i.e., `[Integer]`, and calculates the sum of all digits in the list; returning 0 for the empty list.

   ```
   *Homework2> prob5 [18,1,2,18]
   21
   *Homework2> prob5 []
   0
   ```

Grading

Function	Points
`prob1`:	10
`prob2`:	10
`prob3`:	10
`prob4`:	10
`prob5`:	10
Total	**50**