

Homework 1: Recollecting Basic Haskell

CIS 352: Programming Languages

1 January 2019, Version 0

General notes

- LYH = Miran Lipovacá's *Learn You a Haskell for Great Good*¹
- If you are reading this on a dinky cell-phone screen, be aware that important side-bars are in the right column of this page. So scroll to the right every once in awhile.

Background and Instructions

- This assignment is based on Chapter 1 of LYH.
- Use list comprehensions for these problems, **NOT RECURSIONS**.
- Use the file `hw01.hs` as a starting point for this assignment.
- For each problem, run the QuickCheck tests for that problem. Also, add a few specific tests (non-QuickCheck) of your own.²
- **What to turn in:** (i) your source code,³ (ii) a transcript of your test runs, and (iii) the cover sheet.
- **How to turn it in:** See: <http://www.cis.syr.edu/courses/cis352/reqs.html>

Notes on quickCheck and testRun

QuickCheck⁴ is a Haskell debugging library. For QuickCheck, a *property* is a Haskell function with a type of the form $t_1 \rightarrow t_2 \rightarrow \dots \rightarrow t_n \rightarrow \text{Bool}$. If `convert_prop` is a property, then running

```
quickCheck convert_prop
```

applies `convert_prop` to 100 random inputs. If the function returns True on all the inputs, quickCheck reports:

```
+++ OK, passed 100 tests.
```

If there was a failure (a False), quickCheck reports something like:

```
*** Failed! Falsifiable (after 21 tests and 4 shrinks): 59
```

This means 59 failed the test and `convert` (the function being tested by `convert prop`) has a problem you need to fix.⁵

¹ See <http://learnyouahaskell.com>.

Grading Criteria

- The homework is out of 100 points.
- Each problem is worth 14 points:
 - * 10 pts correctness
 - * 4 your original tests
- You get 2 points for putting your name in the source code file.

² E.g., test that:

```
(isVowel 'x') returns False  
and  
(isVowel 'u') returns True.
```

³ with your name in the comments SVP

⁴ In the past, some people have had problems with QuickCheck loading properly. If you have such problems, let us know ASAP!! There are some fixes.

⁵ The (out-of-date and more-than-you-want-to-know) manual for QuickCheck version 1 can be found at <http://www.cse.chalmers.se/~rjmh/QuickCheck/manual.html>.

The function `testRun` (defined in `hw01.hs`) runs all of the individual QuickCheck tests in `hw01.hs`. So when you have everything working, then evaluating `testRun` should result in something like:

```
*Main> testRun
convert_prop           : +++ OK, passed 100 tests.
vowel_prop             : +++ OK, passed 100 tests.
disemvowel_prop        : +++ OK, passed 100 tests.
smash_prop             : +++ OK, passed 100 tests.
shift_prop_1           : +++ OK, passed 100 tests.
shift_prop_2           : +++ OK, passed 100 tests.
capitalized_prop        : +++ OK, passed 100 tests.
title_prop             : +++ OK, passed 100 tests.
```

If the above isn't the result, you have more work to do.

Your Problems

❖ Problem 1 (Distance: 14 points) ❖

Recall that the Euclidian distance between two points (x_1, y_1) and (y_1, y_2) in the plane is: $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Suppose we have a type synonym

```
type Point = (Double, Double)
```

Implement a Haskell function

```
distance :: Point -> Point -> Double
```

such that `(distance pt1 pt2)` returns the Euclidian distance between the two points.⁶

Use QuickCheck with the property `dist_prop` to test this function.

⁶ The `hw01.hs` file has an example function over Points.

❖ Problem 2 (Testing for vowels: 14 points) ❖

Implement a Haskell function

```
isVowel :: Char -> Bool
```

that tests whether a character is a lower-case vowel, i.e., one of: 'a', 'e', 'i', 'o', and 'u'.⁷

Use QuickCheck with `vowel_prop` to test this function.

⁷ Hint: Use `elem`.

❖ Problem 3 (Disemvoweling: 14 points) ❖

Implement a Haskell function

```
disemvowel :: String -> String
```

which, given a String value, returns that string with all the lowercase vowels removed. For example,

```
disemvowel "mississippi mud pie"
```

should return `"mssssp md p"`.

Use QuickCheck with `disemv_prop` to test this function.

❖ *Problem 4 (Smash: 14 points)* ❖

Implement a Haskell function

```
smash :: String -> String
```

that takes a string s and returns the result of removing all non-letter characters from s and translating each uppercase letter to the corresponding lowercase letter. For example (`smash "Fee, Fie, Foe, & Fum!!"`) would return `"feefiefoefum"`. Defining a helper function is perfectly OK.⁸ Note that in classical cryptography, a message is always smashed (to remove obvious clues) before being encrypted.

Use QuickCheck with `smash_prop` to test this function.

⁸ **N.B.** `isLetter` sadly doesn't do what you want since it is based on Unicode. However, `isLower` and `isUpper` behave as you'd expect.

❖ *Problem 5 (Circular shift cyphers: 14 points)* ❖

A *circular shift cypher* (with shift i) takes a plain text message m and

- (i) smashes m and then
- (ii) replaces each letter with the letter i places down in the alphabet.⁹

E.g., a circular shift of `"Look, a zebra!!"` by 1 results in `"mpplbafcsb"`. Also a shift of `"mpplbafcsb"` by -1 results in `"lookazebra"`.

Implement a Haskell function

```
shift :: Int -> String -> String
```

such that `(shift n s)` does a circular shift of n -places on the result of smashing s . Use list comprehension and `toNum` and `toChar` defined in `hw01.hs`.

Use QuickCheck with `shift_prop` to test this function.

⁹ When we run off the end of the alphabet, we wrap around from the front.

❖ *Problem 6 (Capitalization: 14 points)* ❖

Implement a Haskell function

```
capitalized :: String -> String
```

that takes a nonempty string and properly capitalizes it, i.e., the first character is upper case and the remaining characters are lower case. E.g., (`capitalized "syRaCuSE"`) should return `"Syracuse"`.

Use QuickCheck with `cap_prop` to test this function.

❖ *Problem 7 (Title Capitalization: 14 points)* ❖

Implement a Haskell function

```
title :: [String] -> [String]
```

that given a list of words, capitalizes them as a title. For this problem, that means

- (i) each word over four characters long is capitalized, and

Hint: Think about using a helper function.

- (ii) each word four or fewer characters in length is all lower case—*except* if it is the first word in the input list, in which case it is capitalized.

E.g., `(title ["the", "castle", "of", "wulfenbach"])` should return `["The", "Castle", "of", "Wulfenbach"]`.

Use QuickCheck with `title_prop` to test this function.

Useful functions

```

(&&), (||) :: Bool -> Bool -> Bool
(==), (/=) :: (Eq a) => a -> a -> Bool
(**) :: (Floating a) => a -> a -> a
(:) :: a -> [a] -> [a]

(++): [a] -> [a] -> [a]
abs :: (Num a) => a -> a
chr :: Int -> Char
div, mod :: (Integral a) => a -> a -> a
divMod :: (Integral a) => a -> a -> (a,a)
elem, notElem :: Eq a => a -> [a] -> Bool
head, last :: [a] -> a
init, tail :: [a] -> [a]
isLetter, isLower, isUpper :: Char -> Bool
length :: [a] -> Int
not :: Bool -> Bool
ord :: Char -> Int
maximum, minimum :: (Ord a) => [a] -> a
product, sum :: (Num a) => [a] -> a
sqrt :: (Floating a) => a -> a
toLower, toUpper :: Char -> Char

```

To look up functions that are not explained in LYH, use either:

- <http://www.haskell.org/hoogle>
- <http://hayoo.fh-wedel.de>