Functional Programming I (cs4620) Assignment 3

Pig Latin (Due: October 31. Marks: 5)

1 Introduction

This assignment is about pattern matching and list transformations.

Please remember that your programs should be properly commented. Also please note that all function definitions in your programs should include a proper type signature. Not only is adding them a proper form of documentation but it is also a good exercise.

2 Assignments Details

This assignment is about implementing an English to Pig Latin translator. Please refer to `https://en.wikipedia.org/wiki/Pig_Latin#Rules` for a short introduction to Pig Latin.

Following wikipedia, we use the next two rules for translating words:

1. For a word that starts with a consonant sound, the letters before the initial vowel are moved to the end of the word and then we add "ay" to the end of the word;
2. For a word that starts with a vowel sound, we add "way" to the end of the word.

For this assignment you should implement three versions of a Pig Latin translator function:

Consonant Rule: The first version, which should be named `consonant_translate`, only applies Rule 1;

Vowel Rule: The second version, which should be named `vowel_translate`, only applies Rule 2;

Both Rules: The third version, which should be named `translate_both`, applies both rules.

All these functions transform a single `String` argument to a single `String` output. You may assume the `String` argument consists of letters, punctuation symbols, spaces, and newlines only. Furthermore, you may assume that all letters are lowercase letters. The transformations carried out by the functions should only affect the letters in the words in the input; they should not modify the remaining characters.

Please remember that every `Haskell` program starts by calling the `main` function. The following `main` reads in a line of text, which is returned as a `String`, "assigns" the line of text to a variable `line`, and then prints the `String` that is returned by calling `translate_both line`. Please use this code for your `main`.

```
main :: IO ()
main = do line <- getLine
          putStrLn (translate_both line)
```

You may assume `line` consists of letters, punctuation symbols, spaces, and newlines only.

3 Hints

The following are some hints.

- The function `span`, which is defined in `Data.List`, generalises `takeWhile` and `dropWhile`;
- Think higher-order functions;
- If you implement your translator as a parser that turns a sentence into words, spaces, and punctuation symbols, you can then translate the words using a "word translator," which is a function that takes a word and translates it. Using function composition, you can create a translator for the combined rule from translators for the consonant and vowel rules.

4 Submission Details

- Your program should start with a comment like the following:

```
{-
 - Name: Fill in your name.
 - Number: Fill in your student ID.
 - Assignment: 03.
 -}
```

- Use the CS4620 `moodle` site to upload your program as a single *.tgz* archive called *Lab-3.tgz* before 23.55pm, October 31, 2017. To create the *.tgz* archive, do the following:
  - Create a directory `Lab-3` in your working directory.
  - Copy `Main.hs` (or `Main.lhs`) into the directory. Do not copy any other files into the directory.
  - Run the command '`tar cvfz Lab-3.tgz Lab-3`' from your working directory. The option '`v`' makes `tar` very chatty: it should tell you exactly what is going into the `.tgz` archive. Make sure you check the `tar` command's output before submitting your archive; alternatively, use `tar -t` or `tar --list`.
  - Notice that file names in `Unix` are case sensitive and should not contain spaces.
- Notice that the format is `.tgz`: do *not* submit `zip` files, do *not* submit `tar` files, do *not* submit `bzip` files, and do *not* submit `rar` files. If you do, it may not be possible to unzip your assignment.
- Marks are deducted for poor choice of identifier names and/or poor layout.
- Please should make sure your assignment submission has a `Main` class with a `main` in it. The `main` should be the main thread of execution of the program.
- No marks shall be awarded for scripts that do not compile.