**Assignment 4**

**Overview**

For this assignment, you will develop Java programs to solve several problems. You will also write test cases with which to test the solutions.

If you know some Java already, you may use advanced features that we have not covered in class yet (though it is not necessary to do so). However, you may NOT use any modules or classes from outside java.lang (do not add any "import" statements to your programs).

**Problems 1 - 3  [String and character manipulations]**

***FOR ALL PROBLEMS  you should output your answers, and ONLY THE ANSWERS via the IO module as in previous assignments.Read the attached document on how to use the IO module to output answers.***

**Links to Documentation**

You may find some or all of the following methods of the [Character](http://java.sun.com/javase/6/docs/api/java/lang/Character.html) module useful:

* [isDigit](http://java.sun.com/javase/6/docs/api/java/lang/Character.html#isDigit%28char%29)
* [isLetter](http://java.sun.com/javase/6/docs/api/java/lang/Character.html#isLetter%28char%29)
* [toLowerCase](http://java.sun.com/javase/6/docs/api/java/lang/Character.html#toLowerCase%28char%29)
* [toUpperCase](http://java.sun.com/javase/6/docs/api/java/lang/Character.html#toUpperCase%28char%29)

**Remember that a character is not an object in Java, so you cannot invoke methods on a character in the same way that you would on a string.** You call methods of this module as in this example:

char c = 'a';

char c2 = Character.toUpperCase(c);

System.out.println(c2); // prints 'A'

You may also need to use some or all of the following [String](http://java.sun.com/javase/6/docs/api/java/lang/String.html) methods:

* [charAt](http://java.sun.com/javase/6/docs/api/java/lang/String.html#charAt%28int%29)
* [compareTo](http://java.sun.com/javase/6/docs/api/java/lang/String.html#compareTo%28java.lang.String%29) (alphabetical order, case-sensitive)
* [compareToIgnoreCase](http://java.sun.com/javase/6/docs/api/java/lang/String.html#compareToIgnoreCase%28java.lang.String%29) (alphabetical order, case-insensitive)
* [concat](http://java.sun.com/javase/6/docs/api/java/lang/String.html#concat%28java.lang.String%29) (+)
* [endsWith](http://java.sun.com/javase/6/docs/api/java/lang/String.html#endsWith%28java.lang.String%29)
* [equals](http://java.sun.com/javase/6/docs/api/java/lang/String.html#equals%28java.lang.Object%29) (case-sensitive)
* [equalsIgnoreCase](http://java.sun.com/javase/6/docs/api/java/lang/String.html#equalsIgnoreCase%28java.lang.String%29) (case-insensitive)
* [indexOf](http://java.sun.com/javase/6/docs/api/java/lang/String.html#indexOf%28int%29) (finds a single character)
* [indexOf](http://java.sun.com/javase/6/docs/api/java/lang/String.html#indexOf%28java.lang.String%29) (finds a string)
* [lastIndexOf](http://java.sun.com/javase/6/docs/api/java/lang/String.html#lastIndexOf%28int%29) (finds a single character)
* [lastIndexOf](http://java.sun.com/javase/6/docs/api/java/lang/String.html#lastIndexOf%28java.lang.String%29) (finds a string)
* [length](http://java.sun.com/javase/6/docs/api/java/lang/String.html#length%28%29)
* [startsWith](http://java.sun.com/javase/6/docs/api/java/lang/String.html#startsWith%28java.lang.String%29)
* [substring](http://java.sun.com/javase/6/docs/api/java/lang/String.html#substring%28int,%20int%29) (substring from start, up to but not including end)
* [substring](http://java.sun.com/javase/6/docs/api/java/lang/String.html#substring%28int%29) (substring to end of string)
* [toLowerCase](http://java.sun.com/javase/6/docs/api/java/lang/String.html#toLowerCase%28%29)
* [toUpperCase](http://java.sun.com/javase/6/docs/api/java/lang/String.html#toUpperCase%28%29)
* [trim](http://java.sun.com/javase/6/docs/api/java/lang/String.html#trim%28%29) (trims spaces from ends of string)

**Problem 1**

Write your code in the file PigLatin.java. Write your test cases in assign4-testcases.txt.

"Pig Latin" is a fake language used as a children's game. A word in English is "translated" into Pig Latin using the following rules:

* If the English word begins with a consonant, move the consonant to the end of the word and add "ay". The letter Y should be considered a consonant.
* If the English word begins with a vowel (A, E, I, O, or U), simply add "way" to the end of the word.

(This is a simplified dialect of Pig Latin, of course.)

Ask the user for a word (one string) and output its Pig Latin translation (one string). **You may assume that the input does not contain digits, punctuation, or spaces.** The input may be in any combination of uppercase or lowercase. The case of your output does not matter. Use IO.outputStringAnswer() to output.

**Problem 2**

Write your code in the file WordCount.java. Write your test cases in assign4-testcases.txt.

Write a program that takes a string containing a sentence or a set of sentences, and counts the number of words in the sentence that **meet or exceed** a specified minimum length (in letters). For example, if the minimum length entered is 4, your program should only count words that are at least 4 letters long.

Input the string and the minimum word length (integer), **in that order**, and output the word count (integer). Words will be separated by one or more spaces. Non-letter characters (spaces, punctuation, digits, etc.) may be present, but should not count towards the length of words.

Hint: write a method that counts the number of letters (and ignores punctuation) in a string that holds a single word without spaces. In your main program, break the input string up into words and send each one to your method.

**Problem 3**

Write your code in the file Compress.java. Write your test cases in assign4-testcases.txt.

Run-length encoding (RLE) is a simple "compression algorithm" (an algorithm which takes a block of data and reduces its size, producing a block that contains the same information in less space). It works by replacing repetitive sequences of identical data items with short "tokens" that represent entire sequences. Applying RLE to a string involves finding sequences in the string where the same character repeats. Each such sequence should be replaced by a "token" consisting of:

1. the number of characters in the sequence
2. the repeating character

If a character does not repeat, it should be left alone.

For example, consider the following string:

qwwwwwwwwweeeeerrtyyyyyqqqqwEErTTT

After applying the RLE algorithm, this string is converted into:

q9w5e2rt5y4qw2Er3T

In the compressed string, "9w" represents a sequence of 9 consecutive lowercase "w" characters. "5e" represents 5 consecutive lowercase "e" characters, etc.

Write a program that takes a string as input, compresses it using RLE, and outputs the compressed string. Case matters - uppercase and lowercase characters should be considered distinct. **You may assume that there are no digit characters in the input string. There are no other restrictions on the input - it may contain spaces or punctuation.** There is no need to treat non-letter characters any differently from letters.

**Additional resources for assignment**

* File attachment[IO.java](https://sakai.rutgers.edu/access/content/attachment/67d470ae-bb85-4566-980f-07fb49628dcf/Assignments/0d3b6f16-3a2f-45f8-827a-ce6de24ff20b/IO.java) ( 3 KB; Jan 14, 2015 2:31 pm )
* File attachment[How-To-Use-The-IO-Module-In-Programming-Assignments.html](https://sakai.rutgers.edu/access/content/attachment/67d470ae-bb85-4566-980f-07fb49628dcf/Assignments/d0c6766a-0312-4b06-8231-c69aebfc9361/How-To-Use-The-IO-Module-In-Programming-Assignments.html) ( 4 KB; Jan 14, 2015 2:31 pm )