SCHOOL OF COMPUTER SCIENCE
COURSEWORK ASSESSMENT PROFORMA

MODULE & LECTURER: CMT103 Information Processing in Python; Dr. C Walker

DATE SET: 2nd Nov 2017

SUBMISSION DATE: 29th Nov 2017 at 9:30am

SUBMISSION ARRANGEMENTS:
Your coursework program -- a Python script -- should be submitted by logging into the coursework testbed at https://egeria.cs.cf.ac.uk/ before 9.30am on the submission date.
Make sure to include (as comments) your student number (but not your name) at the top of your script. Also, follow this by any notes (as comments) regarding your submission. For instance, specify here if your program does not generate the proper output or does not do it correctly.

TITLE: Prime Number, Common String, and Top Ly-words

This coursework is worth 30% of the total marks available for this module. The penalty for late or non-submission is an award of zero marks. You are reminded of the need to comply with Cardiff University's Student Guide to Academic Integrity. Your work should be submitted using the official Coursework Submission Cover sheet.

INSTRUCTIONS

Download the following files from Learning Central:

- *cmt103coursework_template.py*
- *numbers.txt*
- *sequences.txt*
- *sense_and_sensitivity.txt*

cmt103coursework_template.py contains a total of 7 to-be-implemented functions each belonging to THREE separate tasks.

The python script also contains code for testing; the three txt files are data set to be used for testing your functions.

To test your implementation using given test files: make sure all the testing files are in the same directory as the python file before you run the Python script.

To test your implementation using your own test files: if you have your testing files, for an instance, *my_nums.txt*, *my_seqs.txt* and *my_book.txt*, you execute the python script (assume it is still called *Cmt103coursework_template.py*) using command line (using either Command Prompt in Wins or Terminal in Mac) which is like this:

```
Cmt103coursework_template.py my_nums.txt my_seqs.txt my_book.txt
```

1. Is It a Prime?

TOTAL 7 MARKS

In this task you implement **Two** functions*:*

1. ***Readnumbers(file_name)*** reads a text file that contains a set of numbers (a line of numbers separated by ‘, ’) and returns a list of the numbers.

   [2 Marks]
2. ***isPrime(num)*** takes an integer as input, examines it and returns *True* if it is a prime number, or *False* if it is not.

   [5 Marks]

numbers.txt has the test data for this task. If both methods are implemented correctly, an execution of the Python script brings the following result:

```
****************************************
Testing Task 1 --- Is It a Prime?
****************************************
Numbers:  [1046527, 1041147, 8356237, 9753423, 9865433, 99733411, 8217,
897933, 67868712, 7676317]

1046527 : Prime
1041147 : Not Prime
```

```
8356237 : Not Prime
9753423 : Not Prime
9865433 : Prime
99733411 : Prime
8217 : Not Prime
897933 : Not Prime
67868712 : Not Prime
7676317 : Not Prime
```

2. *Longest Common SubString*

TOTAL 8 MARKS

In this task you implement **Two** functions:

1. ***readsequences(file_name)*** reads a text file that contains two sequences (separated by '\n') and returns a tuple of the two sequences.

   [2 Marks]

2. ***longest_common_string (st1, st2)*** takes two sequence strings as its inputs and returns the longest common substring that have found in *st1* and *st2*.

   [6 Marks]

sequences.txt is the test data for this task. If both methods are implemented correctly, an execution of the Python script brings the following result:

```
****************************************
Testing Task 3 --- Longest Common Substring
****************************************
The first string: BBEBDEAEBADAAEDCDDBCBACBBCBDDBABBDEECDBAECACEECC
The second string:
CCBADACDCCADDBDABDEDCDDBCBACBBCBDDBABBDEECCACCBDCEBABBBEDC
The longest common substring is EDCDDBCBACBBCBDDBABBDEEC of size 24.
```

3. *Top ly-Ending Words*

Total 15 MARKS

In this task, you will search a text file, finds all the words ending with 'ly' (we call it ly-word), counts their occurrences, gathers the top *10* ly-words and returns them.
The task is split into three functions, and you need to implement the **Three** functions:

1. ***get_words(file_name)*** returns a list of all words that have occurred in the given text file. Be aware that:
   - a word should have multiple occurrences in the list if it has multiple occurrences in the text.
   - The words returned in the list should be of low case and free from any white spaces and punctuations[i] (Except if a punctuation appears in the middle of a word, such as "'" in *I'm*, in this case, we regard it as one word).

   [4 Marks]

2. ***get_dic(words)*** takes the list of words returned by *get_words()* as its input, creates and returns a dictionary, in which each key-value pair is a ly-word and its number of occurrences.

   [4 Marks]

3. ***get_top_ly_words (file, num)*** *takes the* dictionary generated by *get_dic()* as its input and returns a sorted list of tuples. The list must satisfy:
   - Each item in the list is a 2-element tuple: a ly-word and its number of occurrences.
   - It contains the top 10 most frequent ly-words found in the dictionary.
   - The list is sorted in a descending order. The most frequent word is at the front of the list.

   [7 Marks]

sense and sensitivity.txt (by Jane Austin) is the test data for this task. If all functions are implemented correctly, an execution of the Python script brings the following result:

```
****************************************
Testing Task 3 --- Top LY Words
****************************************
+ sense_and_sensitivity.txt has a total of 118573 words.
+ There are 368 ly-words in the file.
+ 'only' and 'hardly' have 281, 65 occurrences respectively.

+ Top 10 ly-Words in sense_and_sensitivity.txt:
     only                281
     really              82
     family              80
     immediately         68
     hardly              65
     certainly           49
     perfectly           43
     entirely            41
     directly            41
     equally             40
```

SUBMISSION INSTRCTUIN

Implement the required functions directly in the script you have downloaded, i.e., *cmt103coursework_template.py*. Rename the name of the script so that it starts with your family name, followed by an underscore '_' and then your student ID. For example, my family name is '*Walker*', suppose my student ID is *12345567*, so I rename my script file to '*walker_12345567.py*'.
You need to **only submit one file**, the python script with the required functions, to https://egeria.cs.cf.ac.uk/.

CRITERIA FOR ASSESSMENT

The functions you have implemented will be tested against different data sets. The score each implemented function receives is judged by its functionality. A correctly functioning function is to be given a full mark.

For a faulty function, the following criteria is to be applied.

Criteria	*Mostly correct. Minor errors in output*	*Major problem. Errors in output*	*Mostly wrong or hardly implemented*
readnumbers() [2]	1	0	0
isPrime() [5]	4	2-3	0-1
readSequence() [2]	1	0	0
longest_common_string() [6]	5	2-4	0-1
get_words() [4]	3	2	0-1
get_dic() [4]	3	2	0-1
get_top_10_words() [7]	5-6	2-4	0-1

Feedback on your coursework will address the above criteria and will be returned in approximately one week. This will be supplemented with oral feedback via lecture, and an exemplar answer will be made accessible via Learning central.

[i] You may like to use string.puncturation provided by Python package. See below for the example:

```
>>> import string
>>> print(string.punctuation)
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```