**AC50002 Programming Languages for Data Engineering**
**Python Assignment**

This assignment consists of a Python programming problem. Your answer should include:

- All relevant source files (i.e. `.py` files); please use a zip file (not a rar file) if there is more than one.
- A *brief* (one or two pages at most) description of the program, indicating how you solved the problem.

The file(s) should be emailed to me (`kjedwards@.dundee.ac.uk`) by 5p.m., Friday 18 December 2015. The files should also be submitted via Blackboard (My Dundee).

## Task

The task is to read a file containing a list of names of some kind, and generate three-letter abbreviations for each of these objects satisfying certain rules, as follows:

- The abbreviations will consist entirely of upper case letters, so for the purpose of finding the abbreviations all the letters can be regarded as upper case.
- Apostrophes (`'`) should be ignored completely. Any other sequences of non-letter characters are also ignored, but split the name into separate words. Examples: (i) "Object-oriented programming" has three words "OBJECT", "ORIENTED", and "PROGRAMMING", (ii) "Moore's Law" has two words "MOORES" and "LAW".
- Each abbreviation consists of the first letter of the name (you can assume that the first character will always be a letter) followed by two further letters from the name, in order. Thus for example, for "Data Engineering", "DTA" and "DEG" are acceptable abbreviations (there are many others), while "DEA" and "ATA" are not.
- Any abbreviation which can be formed from more than one name on the list is excluded.
- Each abbreviation is given a score which indicates how good it is (the lower the score, the better the abbreviation). The scores are assigned as follows: (i) Take the index of each letter, counting from zero at the start of the word it is in, and add these up, (ii) add 10 for each vowel in the abbreviation, unless it is the first letter of its word.

  Thus in the case of "Object-oriented programming", the abbreviation OOP (OBJECT ORIENTED PROGRAMMING) will have score 0 because each letter has index 0 in its word, while the abbreviation OAN (OBJECT ORIENTED PROGRAMMING) has score 24 (0 for the index of O, 5 for the index of A, 9 for the index of N, and 10 for A being a vowel which is not the start of a word).

  Some abbreviations (such as OOG in this example) may be formed in more than one way, with different scores; in this case use the lowest score (3 in this example from OBJECT ORIENTED PROGRAMMING).

The input file should have an extension `.txt`, e.g. `names.txt`. Your program may use several functions, but there should be one main function which, when called, should prompt for the name of the file (e.g. `names`), do the computation, and write the results to the corresponding output file `names_abbrevs.txt`.

Example: Suppose the list consists of just three names, "Cold", "Cool", "C++ Code". Then these have the following possible abbreviations, with the score of each shown:

| | |
|---|---|
| Cold | COL(13), COD(14), CLD(5) |
| Cool | COO(23), COL(14) |
| C++ Code | CCO(11), CCD(2), CCE(13), COD(13), COE(24), CDE(15) |

COL and COD occur in more than one list, so are disallowed, hence the acceptable abbreviations are:

| | |
|---|---|
| Cold | CLD |
| Cool | COO |
| C++ Code | CCO, CCD, CCE, COE, CDE |

In each case we choose from these the abbreviation(s) with the lowest score, giving

| | |
|---|---|
| Cold | CLD |
| Cool | COO |
| C++ Code | CCD |

For this example the input file `names.txt` would contain:

```
Cold
Cool
C++ Code
```

The output file `names_abbrevs.txt` should contain the original names (in their original order) together with the chosen abbreviations, like this:

```
Cold
CLD
Cool
COO
C++ Code
CCD
```

It is possible that there may be no acceptable abbreviation for some word, in which case there should just be a blank line after the word. If more than one abbreviation has the same (best) score, then list them all on the same line, separated by spaces.

For a larger example, you can make use of the file `trees.txt`, which gives a list of British native trees (running the program on this file will produce some quite long lists of acceptable abbreviations).

## Marking Scheme

Marks will be allocated for the following:

- Design (as described in the report) – 10%
- Program
  - Structure, clarity, readability, commenting, user interface – 20%
  - Functionality, ability to carry out what is required – 30%
  - Accuracy, i.e., correctness of operation – 30%
- Testing – 10%

Extra marks may be gained for a particularly neat way of doing something, but will not be gained for extra functionality which was not asked for in the question. Note that marks will be given for the program code and the test strategy even if the program does not work properly, or at all, so you should hand in everything you have done.

## Notes

1. This is an individual assignment. You are free to discuss, in general terms, how you might solve it, but the program which you hand in should be your work alone, except that:

   (a) You are free to make use of any of the example programs given in the AC50002 module, including those in the weekly laboratory solutions.

   (b) You may also use pieces of code obtained from books/Internet, but these must be clearly marked as such in source files. No credit will be given for this code itself, but credit may be given for the way its use enhances the program as a whole.