AC50002 Programming Languages for Data Engineering
Python Assignment

This assignment consists of a Python 3 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. All source file names should start with your surname.
- 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 `m.g.syed@dundee.ac.uk` by 5p.m., Friday 10 December 2021. The files should also be submitted via MyDundee.

Marking Scheme

Marks will be allocated for the following:

- Design (as described in the report) – 10%
- Program
  - Structure, clarity, style, readability, commenting, user interface – 20%
  - Functionality, ability to carry out what is required – 30%
  - Accuracy, i.e., correctness of operation – 30%
- Evidence of useful testing (i.e., what the input was, what the output should have been, and what the output actually was) – 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 design, 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 how you might solve it, but the program which you hand in should be your work alone. In particular, this means that at any time when you are writing the code of your program, you should be doing this without any help from anyone and without any kind of access to the code of another student's program. It is usually very easy to detect when this rule has been broken.

   Note however that the following are completely acceptable:

   (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.

2. You can use any system you wish to write the code, but the code must be in Python 3. Also, in order to avoid difficulties with marking, please ensure that the program will run in the IDLE system by running a main file and then calling a main function called `main()`. Other files can be used, of course, but these should be imported into the main one if necessary.

3. It is not necessary to use regular expressions to do this assignment, though of course you can use them if you wish.

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[1]. 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 score for an abbreviation is the sum of scores for its second and third letters (the first letter is always the first letter of the name, so it does not get a score). These individual letter scores depend on its position in a word of the name and are calculated as follows:

  (i) If a letter is the first letter of a word in the name then it has score 0.

  (ii) Otherwise, if a letter is the last letter of a word in the name then it has score 5, unless the letter is E, in which case the score is 20.

  (iii) If a letter is neither the first nor last letter of a word, then its score is the sum of a position value, which is 1 for the second letter of a word, 2 for the third letter and 3 for any other position, plus a value based on how common/uncommon this letter is in English: 1 for Q,Z, 3 for J,X, 6 for K, 7 for F,H,V,W,Y, 8 for B,C,M,P, 9 for D,G, 15 for L,N,R,S,T, 20 for O,U, 25 for A,I and 35 for E. A list of the values is in the file `values.txt`.

  Thus in the case of "Object-oriented programming", the abbreviation OOP (OBJECT ORIENTED PROGRAMMING) will have score 0 because each letter is the start of its word, while the abbreviation OAN (OBJECT ORIENTED PROGRAMMING) has score 46 (0 for O, 3 for the position of A, 3 for the position of N, plus 25 for the value of A and 15 for the value of N).

  Sometimes an abbreviation (such as DAN as an abbreviation for "Data Engineering") may be formed in more than one way, with different scores; in this case use the lowest score (21 in this example from DATA ENGINEERING).

The input file should have an extension `.txt`, e.g. `names.txt` or `trees.txt`. Your program may use several functions, but there should be one main function `main()` which,

[1] You can assume that all apostrophes are ASCII character 39.

when called, should prompt for the name of the input file (`<input>.txt`), do the computation, and write the results to the corresponding output file which should be called `<surname>_<input>_abbrevs.txt` (so for example if your surname is Jones, and the input file is `trees.txt`, then the output file should be called `jones_trees_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(38), COD(26), CLD(22)
Cool	COO(43), COL(26)
C++ Code	CCO(21), CCD(11), CCE(20), COD(32), COE(41), CDE(31)

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 should contain the original names (which should appear exactly as in the input file) 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 name, in which case there should just be a blank line after the name. 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).