Computer Science 4140
Assignment 3

Implement the American Soundex encoding algorithm described below (source is a Wikipedia entry from 2017, it has been changed since---use this description rather than the current Wikipedia entry)

American Soundex

The Soundex code for a name consists of a letter followed by three numerical digits: the letter is the first letter of the name, and the digits encode the remaining consonants. Consonants at a similar place of articulation share the same digit so, for example, the labial consonants B, F, P, and V are each encoded as the number 1

The correct value can be found as follows:

1. Retain the first letter of the name and drop all other occurrences of a, e, i, o, u, y, h, w.
2. Replace consonants with digits as follows (after the first letter):
    - b, f, p, v → 1
    - c, g, j, k, q, s, x, z → 2
    - d, t → 3
    - l → 4
    - m, n → 5
    - r → 6
3. If two or more letters with the same number are adjacent in the original name (before step 1), only retain the first letter; also two letters with the same number separated by 'h' or 'w' are coded as a single number, whereas such letters separated by a vowel are coded twice. This rule also applies to the first letter.
4. If you have too few letters in your word that you can't assign three numbers, append with zeros until there are three numbers. If you have more than 3 letters, just retain the first 3 numbers.

Using this algorithm, both "Robert" and "Rupert" return the same string "R163" while "Rubin" yields "R150". "Ashcraft" and "Ashcroft" both yield "A261" and not "A226" (the characters 's' and 'c' in the name would receive a single number of 2 and not 22 since an 'h' lies in between them). "Tymczak" yields "T522" not "T520" (the chars 'z' and 'k' in the name are coded as 2 twice since a vowel lies in between them). "Pfister" yields "P236" not "P123" (the first two letters have the same number and are coded once as 'P'), and "Honeyman" yields "H555".

It is much easier to implement the following equivalent algorithm (no need to return to the original name as in step 3 above).

1. Save the first letter. Remove all occurrences of ‘h’ and ‘w’ except first letter.
2. Replace all consonants (include the first letter) with digits as in step 2 above.
3. Replace all adjacent same digits with one digit.
4. Remove all occurrences of a, e, I, o, u, y except first letter.
5. If first symbol is a digit replace it with letter saved in step 1.
6. Append 3 zeros if result contains less than 3 digits. Remove all except first letter and 3 digits after it (This step same as step 4 in explanation above).

Program Details

Your program should take in a raw text file, read it in line by line, break each line into tokens. Tokens that have non alphabetic characters should be output as is. Tokens that consist only of letters should be transformed according to the algorithm. Further details are the following.

- A starter template for the file *soundex.py* is provided. It contains the code for processing the text file. The name of the file is supplied on the command line. Thus, a sample execution of the program would be the following command line.

  python3 soundex.py input.txt

- The code you write will actually just be the code needed to implement the function *wordmap(token)* that takes a token and returns the token transformed according to the rules given.

- Preceding the definition of the *wordmap* function, you may wish to initialize some global strings that can help in constructing regular expressions to be used in pattern detection.

- The most elegant approach for handling the mapping of consonants to digits is to use a dictionary to define the mappings. I would suggest you create a list of tuples where each tuple is of the following form *(<list of letters>, <char>)* (e.g. *([‘d’,’t’],’3’)*). Then with a simple nested loop you can build the dictionary from the tuple list. Specifying the mapping as a list of tuples is convenient for readability.

Program Deadlines

February 19, 11:00 P.M.

Deadline for submitting completed programs. Use *hw3* as the *<assignment_ID>*.

Program Evaluation

The assignment is worth 40 points. Your grade on this assignment will be weighted as follows:

Code Correctness	70%
Coding Style and Documentation	30%