CSCI 1133
Exercise Set 6

You should attempt as many of these problems as you can. Practice is *essential* to learning and reinforcing computational problem solving skills. We encourage you to do these without any outside assistance.

Create a directory named `exercise6` and save each of your problem solutions in this directory. You will need to save your source files in this directory and push it to your repository in order to get help/feedback from the TAs. This requires that you follow a rigid set of naming requirements. Name your individual Python source files using "`ex6`" followed by the letter of the exercise, e.g., "`ex6a.py`", "`ex6b.py`", "`ex6c.py`", etc.

Automatic testing of your solutions is performed using a GitHub agent, so it is necessary to name your source files precisely as shown and push them to your repository as you work.

A. **Strings and Lists**

Write a Python module that contains a single pure function: `wordlist(prose)`, that takes a single string argument containing some English prose and returns a Python list containing just the (unique) words, one word per list element. The words in the returned list should all be lower case. The prose contains an unknown number of words separated by spaces, commas, periods and semicolons (no other punctuation). Also note that individual words can be separated by any number of spaces. Use any string methods as appropriate.

Example:

```
>>> wordlist('It was the best of times, it was the worst of times; It was the age
of wisdom,         it was the age of foolishness')

['it','was','the','best','of','times','worst','age','wisdom','foolishnesss']
```

B. **Word Separation**

Write a Python program that contains two pure functions `separate(phrase)`, and `combine(phrase)`

First, construct the function `separate(phrase)` that takes a single string argument containing an English phrase in *camelcase* (where all the words appear together without spaces and each separate word starts with an uppercase letter). The function should convert the *camelcase* phrase to a string in which the words are separated by spaces and only the first word of the phrase starts with an uppercase letter. Use any string methods as appropriate.

Example:

```
>>> separate('StopAndSmellTheRoses')
Stop and smell the roses
```

Now write a second pure function: `combine(phrase)` that will do the inverse of the previous problem, i.e., take a phrase and convert it to *camelcase*. Use any string methods as appropriate.

Example:

```
>>> combine('Stop and smell the roses')
StopAndSmellTheRoses
```

Your main function should input phrases, convert them to camelcase and then use the separate() function to recover and display the original phrase.

C. **English to Pig Latin Translator**

Write a pure function named `igpay` that will translate an English word-phrase into Pig Latin. Pig Latin is a fun word game in which words are translated according to the following "rules":

1. For any word that begins with one or more consonants (y is considered a consonant): move the consonants to the end of the word and append the string 'ay'.
2. For all other words, append the string 'way' to the end.

For example, the phrase "Can you speak Pig Latin?" would be translated:

"Ancay ouyay eakspay Igpay Atinlay?"

Or, "An apple a day keeps the doctor away" would be translated:

"Anway appleway away ayday eepskay ethay octorday awayway"

Your function should take a single string argument and return a converted string. It should also preserve the capitalization of individual words (see examples) and the following punctuation: commas, exclamation points, periods, hyphens and question marks. All other punctuation characters can be considered consonants that are part of the word. For this exercise, you may assume that individual words will be separated by one or more spaces or hyphens. You are free to use any standard Python string methods in your solution.

[Hint: you will find this much easier if you first define a separate function to locate the initial vowel of any word, and then use that function in your solution]

D. **Telephone Number Translator**

You often see telephone numbers that are written as words to help you remember: "1-800-GET-WELL" or "1-888-Beer2Go. The letters are mapped to numbers according to an international standard as follows:

ABC = 2
DEF = 3
GHI = 4
JKL = 5
MNO = 6
PQRS = 7
TUV = 8
WXYZ = 9

Write a Python program that will do the following:

- Prompts the user and inputs a telephone number string in 'character' form from the console
- Calls `telTranslate` to translate the number to 'digit' form
- Displays the result on the terminal display
- Includes a loop that will continue this process until the user enters a null string.

Your program needs to include two separate pure functions: `charNumber(char)` that will accept a single character in the range 'A'-'Z' and return the integer value for that letter. If the argument is not in range, return a null string, and a second function named `telTranslate(phrase)` that will accept a single string argument containing a telephone number in 'character' form and return a *string* consisting of the telephone number digits with '-' separators as appropriate. The function must include a call to the `charNumber` function to convert each *alphabetic* character to a digit.

Example:

```
enter a telephone number: 1-800-GET-WELL
number is: 1-800-438-9355

enter a telephone number: 1-800-Beer2Go
number is: 1-800-233-7246

enter a telephone number: 1-800-7358721
number is: 1-800-735-8721

enter a telephone number:
```