CSCI 1133
Exercise Set 9

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 `exercise9` 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 "`ex9`" followed by the letter of the exercise, e.g., "`ex9a.py`", "`ex9b.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. **Counting Keywords**

Write a Python program that will read any Python source program file and print out a frequency count of the *reserved* words in the file. You can find a list of the Python keywords on the Python.org website:

```
docs.python.org/3/reference/lexical_analysis.html#keywords
```

Your program should print each keyword and its associated count, one per line, in *alphabetic order*. Do not list keywords that appear zero times in the file. Do not import or use any Python modules in your program.

Your program should do the following:

- Prompt the user and input the name of a Python source code file (`.py`)
- Read the contents of the file and construct a *dictionary* consisting of the keywords and their associated counts
- Print a list of all keywords and their associated counts in alphabetic order of keywords (include only those with non-zero counts)

Example:

```
Enter the filename: junk2.py
Keyword frequency in alphabetic order:
  def       4
  for       4
  if        3
  in        6
  return    3
```

B. **Fun with Excel**

Write a Python program that will read an Excel spreadsheet in `.csv` format and output the contents to the display, one row at a time without the commas. The values in each row should be aligned vertically:

```
Jan   Feb   Mar   Apr
  0    21    42    11
  1     4     9    22
       69   103   200
```

C. **Zipf's Law** (adaped from *Introduction to Computing Using Python*, Perkovic, problem 6.37)

Zipf's Law (George Kingsley Zipf, 1902-1950) holds that the usage frequency of any word is inversely proportional to its ranking:

$$P_n \sim C * 1/n$$

Where C is a proportionality constant and P_n is the *frequency* of the n^{th} most frequent word (defined as the number of times it occurs in a text divided by the total number of words in the text).

Write a Python program that will solicit a filename from the user, read all the words in the text file and verify Zipf's Law by printing an estimate of $C = P_n * n$ for the first 20 words in the file. Ignore capitalization and punctuation:

Example:

```
% python3 ex9c.py
input filename: frankenstein.txt
0.055731955
0.079047708
0.113270715
      .
      .
      .
```

D. **Flight Data Filter**

A *filter* is a program that reads each record of an input file and creates a new output file containing only those records that match some qualifying criteria. For this assignment, you will construct a short Python *filter* program that will process real-world airline flight-routing data and produce a file containing only those records that represent flights arriving to and/or departing from a particular airport.

Use your web browser to view the following URL:

http://openflights.org/data.html

This is a website that maintains information on airline flights. Browse the site, locate the section entitled: "*Route Database*" and download the `routes.dat` file (~2MB). This file describes all worldwide airline flight segments between various airports. You should also review the text that describes the content and format of the file.

Write a Python program that will process this file and produce a smaller file containing only those flights that arrive at, or depart from, a designated airport. Your program should solicit the name of the input file, the name of the (filtered) output file and the IATA letter code for the selected airport. For example, Minneapolis is MSP and San Francisco is SFO, etc.

Requirements:

Your program must do the following:

- Solicit the name of the input route-data file from the user. If the file does not exist or fails to open, continue to solicit the file name a maximum of 3 times and then terminate the program
- Solicit the name of the output (filtered) route-data file from the user. If the file exists, then ask the user if he/she would like to overwrite the existing file. If not, then terminate the program.

- Solicit an IATA airport code from the user (e.g., MSP, SFO, etc.). Your program should accommodate entry in either upper or lower case.
- Process the input file and produce an output file consisting only of the records of those flights that arrive and/or depart from the indicated airport.

Constraints:

- Do not import/use the Python `csv` or `shutil` modules. All other standard Python modules are acceptable, including `os`
- All files must be properly opened and closed.

Examples:

```
Airport Routing Filter
Enter the source file name: routes.dat
Enter the output file name: mspdata
Enter airport symbol: msp
Finished


Airport routing filter
Enter the source file name: mickeymouse
File not found.  Reenter: routes.dat
Enter the output file name: mspdata
File exists... overwrite? (y/n): y
Enter airport symbol: msp
Finished
```