COP 2930 - Individual Programming Assignment #11

Due date: Please consult WebCourses for your due date/time

Objectives

1. Practice reading in input from files.
2. Practice using dictionaries.
3. Practice writing output to files.

Problem: Reporting COVID-19 Test Results (report.py)

Each COVID-19 testing site must report a summary of their results to the state. In this problem, we'll investigate the issue of coalescing multiple data files into one, where we report a single result per individual. (In this example, what we are doing different is we are keeping track of names and we are only recording a single testing result per person, even if they might have taken more than one test. These changes are simply to give you practice with the skills that I'd like to give you practice with.)

A testing site records several text files of data. Each text file has a summary of some of the COVID-19 tests given on a particular day. The problem is that sometimes results from the same test get recorded in multiple files and another problem is that sometimes a person actually has more than one test in a day. For each individual who took at least one test, we would like a single record: either a "positive" or "negative". To be on the cautious side, if any entry for an individual shows a "positive", we should record their result as "positive". (So, for the purposes of this problem, 7 negatives and 1 positive is a yes, but 1 negative is a negative.)

Your program should ask the user if they have any more files to process. If the user has a file to process, your program should prompt the user for the name of the file and then process that file. This process should repeat until the user says they have no more files to process.

At this point, ask the user which file they would like their coalesced results written to.

Then, write the results to that file. In particular, this file should store the results in alphabetical order by name, one result per line. Each line should have the patient's name, followed by a space, followed by either the string "positive" or "negative".

Input File Specification

The first line of the input file will contain a single positive integer, *n*, representing the number of test results in the file. This is followed by 2*n* lines. Each pair of lines stores information about one test result. The first of a pair of lines stores the name of the patient, in the format, last, first. The second line either stores the string "positive" or "negative". Your program should read in the whole name as a single string and treat two names the same only if the two strings are identical. Here is a sample input file:

```
5
White, Karen
negative
Smith, Joe
negative
King, Ellen
positive
Smith, Joe
negative
Smith, Jim
negative
```

Output File Specification

In the output file, the first line should contain a single positive integer, representing the total number of entries.

The following lines should contain the names of the patients in sorted order. For each patient, there should be a space following their name, followed by either "positive" or "negative". Put the former if any of that patient's recorded test results are positive, and put the latter, otherwise.

If the only file entered was the sample file above, then the stored output file should be:

```
4
King, Ellen positive
Smith, Jim negative
Smith, Joe negative
White, Karen negative
```

Input Specification

Note: It is guaranteed that whoever uses your program will adhere to these specifications. This means that you do NOT have to check for them!

The user will enter all valid input. Namely, when asked if they want to enter another file, they will properly enter either "yes" or "no" (without quotes). When asked to enter a file name, they will enter a file name that exists in the current directory and the file will be in the appropriate format.

At the end when the user is asked to enter an output file name, they will enter a valid name of a file that doesn't exist in the current directory.

Output Specification

Follow the sample given below. Most of the output to the screen is simply acknowledgements of completing tasks.

Output Samples

Below is one sample output of running the program. **Note that this sample is NOT a comprehensive test.** You should test your program with different data than is shown here based on the specifications given above. In the sample run below, for clarity and ease of reading, the user input is given in *italics* while the program output is in **bold**. (Note: When you actually run your program no bold or italics should appear at all. These are simply used in this description for clarity's sake.)

Sample Run

```
Would you like to upload another file(yes/no)?
yes
What is the name of the file?
data01.txt
File data01.txt has been uploaded.
Would you like to upload another file(yes/no)?
yes
What is the name of the file?
data02.txt
File data02.txt has been uploaded.
Would you like to upload another file(yes/no)?
yes
What is the name of the file?
data03.txt
File data03.txt has been uploaded.
Would you like to upload another file(yes/no)?
no
Which file would you like the final results stored?
finalresults.txt
Your results have been stored.
Thank you for using the test data accumulator program!
```

Restrictions

Please IDLE 3.6 (or higher) to develop your programs. Write a single program named **report.py** which solves the designated problem and submit this file via Webcourses.

Your program should include a header comment with the following information: your name, course number, assignment title, and date. Also, make sure you include comments throughout your code describing the major steps in solving the problem. For the second program, please leave our tests in the file for ease of grading.

Grading Details

Your programs will be graded upon the following criteria:

1) Your correctness and filling in the given function prototype appropriately.

2) Your programming style and use of white space. Even if you have a plan and your program works perfectly, if your programming style is poor or your use of white space is poor, you could get 10% or 15% deducted from your grade.

3) Compatibility to IDLE.