CS2204
Project No. 7

Purpose:
Gain experience in using Python to manipulate CSV files, which are commonly used to share data.

Background:
At the end of each semester, I need to perform the tedious task of computing grades for all the students in all my classes. I download a CSV file from Oak containing the scores students earned on all work items. I then convert that file into an Excel spreadsheet, into which I insert formulas that I have developed over the years to compute weighted averages, final grades, and class ranks. From this file, I then report final grades to the registrar.

The Assignment:
In this assignment, we are going to create a program that will perform all the needed computations without the need for Excel. You will be provided with a starter Python program and a CSV data file. The data file is one created by Oak for an actual class I have taught; only the personal identification information has been replaced with generic data.

You are being provided with the following:
1. project7.py: a driver program for this project
2. Gradebook.py: a definition of a Gradebook class. You will modify this file to add the necessary functionality to the class.
3. grades.csv: the file containing the input CSV data whose format is described below

Your task will be to add code to the Gradebook.py file to perform the computations necessary to determine each student's final grade. The steps below describe the computations.

Input File Format
The input file is a CSV file meeting the following specification:
- Columns are organized as follows:
  - Columns 1-3 are student identification (last name, first name, and vunetid)
  - The next *n* columns present *n* homework assignments (the number *n* changes from semester to semester, so it is not known ahead of time)
  - The last 4 columns represent the exams, in order of exam1, exam2, exam3, and the final exam
- Rows are organized as follows:
  - The first row contains column headers. Each column has a unique header, but the homework headers share a common prefix ("proj") and the exam headers share a common prefix ("exam").
  - The second row contains the points possible for each graded column (this row may contain data in non-graded columns, which is maintained in the output file but not processed in any manner).
  - Afterward, there is one row of data per student
  - Some graded entries in a student row may be empty [entry contains either an empty string or a single blank character]
    - Represents a graded item not turned in by the student
    - The student receives a zero grade for that item

Step I:
Create a new project for this assignment. Once the project is created, copy the two supplied Python files into the src directory of the project and add them to the project. Also copy the supplied CSV file into the project directory (one directory above the Python files).

Take a moment to review the supplied code. The main method in the project7 driver simply creates a Gradebook object from the CSV file, and then calls two methods on that object (one to compute the desired data and the other to write the output CSV file). It also echoes the computed data to the screen for a quick sanity check.

The Gradebook.py file contains headers for several methods. You will write these methods by following the steps below. One method is provided for you: the computeGrades() method is the driver for computing grades. You are encouraged to write your own helper functions to modularize your code.

Step II:
Implement the `__init__()` constructor function. The responsibility of the constructor is to open the specified CSV file, and read all its data into an internal structure (a list of lists, where each sublist represents a row). The constructor is responsible for performing some pre-processing of the raw data to ensure consistency. All rows should be the same length as the first row. If you encounter a row longer than the first row, it should be truncated to the correct length (and an appropriate warning message should be printed to the screen). Any row shorter than the first row should be padded out to the correct length by adding '0' entries at the end of the row. Any empty data fields in the graded portion of a student record should be replaced with '0' entries. An empty data field is an empty string or a string with a single blank.

Step III:
Implement the `compute_weighted_average()` function. This function computes the weighted average for each student based upon the scores they earned and on the weightings specified in the course syllabus. Averages are computed by dividing the points earned within a grading category by the points possible for that category. The categories are homework assignments, exam #1, exam #2, exam #3, and a final exam (exam #4). The weightings are 45% of final grade is based on homework, 10% is based on exam#1, 10% on exam #2, 15% on exam #3, and 20% on the final.

Add each student's weighted average to their information by appending it to the end of their row (thus creating a new column). Append an appropriate column header to the first row.

This function returns a list of all the weighted averages in student order (i.e., the first weighted average in the list corresponds to the first student in the Gradebook). Even though the main driver ignores this return value, you must return it.

Step IV:
Implement the `compute_class_rank()` function. This function computes where each student ranks in the class based upon the weighted averages. The student with the highest weighted average is ranked #1. If two students score the same weighted average, they should be ranked as a tie [i.e., if two students have the third highest weighted average, they should both be ranked #3 and the next student should be ranked #5].

Add each student's rank to their information by appending it to the end of their row (thus creating a new column). Append an appropriate column header to the first row.

This function returns a list of the rankings in student order (i.e., the first ranking in the list corresponds to the first student in the Gradebook). Even though the main driver ignores this return value, you must return it.

Step V:
Implement the `assign_grades()` function. This function computes the final grade for each student based upon their weighted average. Grades are computed on a straight 90-80-70-60 scale; e.g., all students with a weighted average >= 90 earn an 'A', all students with a weighted average <90 and >=80 earn a 'B', etc.

To be nice to the students and to avoid hearing them beg for better grades, we always round the weighted average up to the nearest integer. We do not change the values of the weighted average in the Gradebook when we do this rounding.

Add each student's grade to their information by appending it to the end of their row (thus creating a new column). Append an appropriate column header to the first row.

This function returns a list of the grades in student order (i.e., the first grade in the list corresponds to the first student in the Gradebook). Even though the main driver ignores this return value, you must return it.

Step VI:
Implement the `write_grades()` function. This function writes the Gradebook information out to the specified CSV file. When writing the data to the file, we want the weighted averages to be rounded to two decimal digits. You may either change the values in the Gradebook object or use format commands to alter the writing of the data. Note that the class rankings and final grades are determined on the full floating point precision of the weighted average and not the rounded value.

Step VII:
Test your program with the supplied data file and with other data files of your own making (that obey the input file format specified earlier).

Step VIII:
For grading, simply submit your updated `Gradebook.py` file.

Sample execution:
Given the supplied CSV file, here are the first six that should appear on the screen:

```
Wt.Avg  Rank Grade
94.42%   21    A
96.48%    9    A
87.97%   56    B
93.29%   32    A
84.80%   65    B
```

And here are the first six student lines (after the column header and points possible lines) that should appear in the output CSV file:

```
A,A,A,30,30,30,19,26,40,28,40,26,26,36,20,29,28,28,5,5,5,5,5,5,5,95,97,96,90,94.42,21,A
B,B,B,30,30,30,19,30,40,33,40,28,30,38,20,36,28,30,5,5,5,5,5,5,5,99,89,94,92,96.48,9,A
C,C,C,30,30,23,14,30,37,24,21,22,24,38,20,17,20,25,5,5,5,5,5,5,5,94,84,90,97,87.97,56,B
D,D,D,29,30,30,20,30,40,28,38,15,28,38,20,28,30,30,5,5,5,5,5,5,5,77,91,95,98,93.29,32,A
E,E,E,28,30,30,19,30,40,29,40,26,30,37,20,30,30,30,5,5,5,5,5,5,5,83,75,76,68,84.8,65,B
F,F,F,28,30,30,19,30,40,23,40,28,28,40,18,36,30,28,5,5,5,5,5,5,5,93,87,96,94,95.11,16,A
```