NYU, Tandon School of Engineering
CS-UY 1121 Problem Solving via Programming — Spring 2023

Homework #1
Due: 5:00 pm, Tuesday, February 7th, 2023

Note: Do not use any language construct not taught in lecture sections. For this homework, that includes file i/o, exception handling, dictionaries, and object orientation. For this assignment you can use everything taught in CS 1113. Not sure about a library or language feature? Ask me.

Submission instructions

1. You should submit your homework on Gradescope.
2. For this assignment you should turn in a single .py file.
3. There is no requirement to name the file(s) in any particular pattern or manner.
4. Each Python file you submit should contain a header comment block as follows:

```
# Author: <Your name>
# Assignment / Part: HW1
# Date due: 2023-02-07
# I pledge that I have completed this assignment without
# collaborating with anyone else, in conformance with the
# NYU School of Engineering Policies and Procedures on
# Academic Misconduct.
```

Data Analysis Using Nested Lists

In this problem you are provided with a large amount of data (large considering what you have programmed previously (but not large as in a professional environment setting).

Our data for this homework is a list of students, some of their academic information, and a list of 3 classes (and their grades). We use nested lists to store the information. Let's take a look at a small example and build on it. First, here's some data for a single student.

Our data is organized as follows:

- Student first name, last name, and N-number,
- a sublist containing their academic (undergraduate) year (freshman, sophomore, junior, senior, or "other"), their Tandon major, their current GPA, and their current number of credits achieved,
- then, a list of three classes in which the student is currently enrolled. Each class contains three lists of grades (in this order): quizzes (3), homeworks (7), and tests (3)

```
['Mason', 'James', 'N81649046',
     ['other', 'Business and Technology Management', 3.557, 55],
     ['phy1222',
          [[92, 89, 93],
           [92, 80, 76, 79, 89, 77, 100],
           [23, 27, 19]]
     ],
     ['phy1223',
          [[50, 40, 60],
           [2, 29, 3, 3, 19, 26, 6],
           [66, 64, 72]]
     ],
     ['phy2144',
           [[26, 21, 17],
           [81, 60, 85, 54, 79, 91, 81],
           [19, 14, 30]]
     ]
]
```

Keep in mind, the above data is for one student (represented as a single list, with multiple sub-lists). Of course, your input is really many more students, and thus they will be provided in a list. So a representation of an input list for this data would be something like:

```
[ [student1…], [student2…], …]
```

Note that the data was generated by a Python program that randomly created all of the data. There may be issues of duplication in the data, and if so, you should reach out to the professor on how to deal with it. For example, I know that there is at least one occurrence where a single student has 3 courses, but one of the course names is duplicated. For a case like this, we will ask you to continue processing the duplicate course as if it were not a duplicate.

The overall data set will be provided as a list of data that you can copy/paste into your programs (since we are not yet reading input from a file). You can access the data here: https://nyu.box.com/s/i7vgqtbosqwlttg8yf1a1b37ocj1yawi

Program Goals

Your solution will essentially access the data provided and calculate a number of results:

1. A listing of all student names (first and last) and gpas, grouped by their academic year.

2. For each student, calculate their final course numeric grade as follows. The average of the quizzes is worth 10% of their overall grade. The average of the homeworks (after you drop the lowest grade) is worth 15% of the overall, and the three tests are worth 20% (1st test), 25% (2nd test), and 30% (3rd test). This

portion of the output should be sorted descending by the final numeric grade (hint: look into the sort method used with lists and define a function that specifies the sort key).

3. Find and print the most popular first name from the data. Hint: sorting the list might be helpful here.

Program Requirements

The program should be written in one python file (your choice on the name). There should be several functions for this solution as follows:

- **[5 points]** `main` - The main function should be called by the Pythonic-way to identify the file where the program starts (`if __name__ == "__main__":`).
- **[10 points]** `load_data` - this function should return the input data (as a list). Since you can copy/paste the data from the provided input file, this function should be pretty small. Later, we'll learn how to read the data file from the web directly, and then copying/pasting won't be necessary.
- **[35 points]** `print_roster_by_year` - uses the data list to build and print a roster of students (see Program Goal #1)
- **[25 points]** `print_course_grades` - uses the data list to build and print the final course grades for each course for each student (see Program Goal #2)
- **[25 points]** `print_most_popular_first_name` - identifies the most frequent first name in the dataset. If there is a tie, then print all names (see Program Goal #3)

Notes

- Feel free to introduce other "helper" functions in addition to the required functions above. Good function design should be in the range of 20 lines or smaller. As you develop the required functions to solve the problem, you may find it helpful to add smaller functions ("helper functions") to help decompose the larger functions into smaller ones.