Multi-Paradigm Programming
Repeat Project (100%)

Dominic Carr

For this assessment you are asked to build a GPA calculator for students in a fictional university. You will be supplied a CSV file with names and the grades they have achieved in various modules (MPPSample.csv). This will be used to calculate GPA. You should add additional data to this CSV for the purposes of testing.

You will need to convert the percentage grade (i.e. a grade out of 100%) given in the CSV to a letter grade and it's corresponding value on the GPA scale. The GPA scale should run from 4.2 down with + and - grades. Each module has equal weighting for GPA calculation.

An A+ is worth 4.2, an A 4, and A- 3.8 and so on down to F. You can decide what corresponds to an A+ for example you could choose greater than 90%.

There should also be a live mode where the user can enter a set of marks and modules from the command line and have the program calculate the GPA.

The final output of the program should show the GPA of each student, their highest scoring module, their lowest scoring module, standard deviation, median value, how far from the next highest GPA they were (if not at 4.2), and the letter grade of each module result.

Notes

- The above described functionality should be completed in Python and C. This is to be done in a procedural programming style. Structs should be used in the C implementation. In your report mention the best practices you have followed.
- The second part of the assessment involves replicating the functionality in **Java or Python**. This must be done in an *Object Oriented manner*. The OOP program should have appropriate object representations, and a division of responsibility among several classes. **This should be justified in your report.**
- You must complete a short report, **around 3-5 pages**, which **compares the solutions achieved** using the procedural approach and the object oriented approach. This is NOT intended to be a report on the paradigms themselves but rather on how applying the paradigms worked in your specific solution.
- The live mode, and the input files, should have the exact same behaviour in ALL implementations.
  - For example I should be able to use the Python implementation in the same way as the C one i.e. same CSV files, and the same process in live mode.
  - The "user experience" of each implementation should be **identical**.

Marking Scheme

- Procedural Python Program (25%)
- Procedural C Program (25%)
- OOP Java/Python Program (40%)
- Report (10%)
  - Spelling, grammar & writing (2.5%)
  - Analysis of similarities and differences (7.5%)