Software Assessment - Report Cards

In this assessment, you will write a script to generate a text file containing the "report card" of all students in the database.

You will be given an example input and an example output. Your program should be able to handle different inputs of various sizes. Your program must read the csv files and iterate through them.

This assessment will be evaluated based on the following criteria:

- Completion: Did you complete all the steps? Does your solution work for different inputs? **Do not hardcode the output given these example inputs - we want your solution to be able to handle different inputs in the same format.**
- Fundamentals: Is your code efficient? Do you use the appropriate data structures to store information?
    - We are *not* looking for a solution that just stores all the data in a database and performs a query from the database. We want you to use programming concepts such as lists and maps as opposed to database or data analytics technologies. Try to avoid nested for loops in this problem.
- Code Organization and Quality: How organized is your solution? Is the code easy to read?
- Testing: Did you write some tests for the solution? How is the coverage?
- Speed: We are looking for solutions that are completed within an appropriate time frame (1 day is an appropriate time frame).
- Communication: We are looking for candidates with strong communication skills. This will be evaluated based on your Readme.md file.

Inputs

You will be given four files as an input to your program. The description of each file is written below. Here is a compressed folder with a simple example input.

courses.csv

This file contains the courses that a student takes. Each course has a **unique id**, a **name**, and a **teacher**.

students.csv

This file contains all existing students in the database. Each student has a **unique id**, and a **name**.

tests.csv

This file contains all the tests for each course in the courses.csv file. The file has three columns:

- **id:** the test's unique id
- **course_id:** the course id that this test belongs to
- **weight:** how much of the student's final grade the test is worth. For example, if a test is worth 50, that means that this test is worth 50% of the final grade for this course.

The sum of all the weights of all tests in a particular course should add up to 100. You are allowed to throw errors in your report card generation script if this is not the case.

marks.csv

This file contains all the mark the student received for all the tests that they have written.

The file has three columns:

- **test_id:** the test's id
- **student_id:** the student's id
- **mark**: The percentage grade the student received for the test (out of 100)

Note: **Not all students are enrolled in all courses** – this can be determined by the marks that they receive. All students should have completed (taken every test for) each course they are enrolled in. **If a student takes no test in a course, then this student is not enrolled in that course.** If there are students that have not completed a course, you can throw an error in your report card generation script.

Output

Given the example input, here is the text file that your program should generate. Your goal is to write a program that generates a text file with the following characteristics:

- All student report cards – the order of the students should be based on student id

Here is an example report card:

```
Student Id: 1, name: A
Total Average:      72.03%

     Course: Biology, Teacher: Mr. D
     Final Grade:   90.10%

     Course: History, Teacher:  Mrs. P
     Final Grade:   51.80%

     Course: Math, Teacher:  Mrs. C
     Final Grade:   74.20%
```

- The first line should contain the student id and the student's name
- The second line is the average of all the courses the student is enrolled in
- Below that, a listing of all courses and the student's grade in each course (this will be determined by the mark they get in each test, and how much each test is worth)
- The courses are ordered by course id

Readme

Along with your submission, write a Readme.md file that:

- How to run the application:
    - The command we should use installing your dependencies, e.g: *npm install, pip install -r requirements.txt*. You can assume we already have your programming language set up in our environment, as well as any common package managers (npm, yarn, bundle, pip, etc)
    - The command we should use for actually running the application, e.g: *npm start, java Main, python manage.py serve*
- A brief (three sentences) high-level description of your project, written for a non-technical person. This will help us assess how well you can communicate.

Submission Details

Please submit your code in a compressed folder (.zip, .sitx, .7z, .rar, and .gz) on the Hatchways platform. The max submission size is 5MB.

Please include a Readme.md that describes how to run the application. Do not submit any built folders, since the compressed folder will be too large.

If your submission is too big (although this should not be the case if you follow the steps above), and you can't figure out how to compress, you are welcome to email your solution to hello@hatchways.io.

Please include your name, and use the email you signed up with the Hatchways platform. Use the subject line “Backend Assessment Submission”.

Public Repositories

Please avoid posting your solution to a public repository. We understand that you may want to share projects you have worked on, but many hours go into developing our tools so we can provide a fair skills evaluation.

If you would like to keep a similar version of the assessment on a public repository to showcase your skills, then please do the following:

- Remove all references to Hatchways in the assessment
- Do not use any Hatchways APIs (replace them with an API of your own)