Homework 09

Re-submit Assignment

Due Apr 1 by 10pm **Points** 100 **Submitting** a file upload

You've been hired by Stevens Institute of Technology to create a data repository of courses, students, and instructors. The system will be used to help students track their required courses, the courses they have successfully completed, their grades, GPA, etc. The system will also be used by faculty advisors to help students to create study plans. We will continue to add new features to this solution over the coming weeks so you'll want to think carefully about a good design and implementation.

Your assignment this week is to begin to build the framework for your project and summarize student and instructor data. You'll need to download three ASCII, tab separated, data files:

1. **students.txt** (available from Canvas)

- Provides information about each student
- Each line has the format: CWID\tName\tMajor (where \t is the <tab> character)

2. **instructors.txt** (available from Canvas)

- Provides information about each instructor
- Each line has the format: CWID\tName\tDepartment (where \t is the <tab> character)

3. **grades.txt** (available from Canvas)

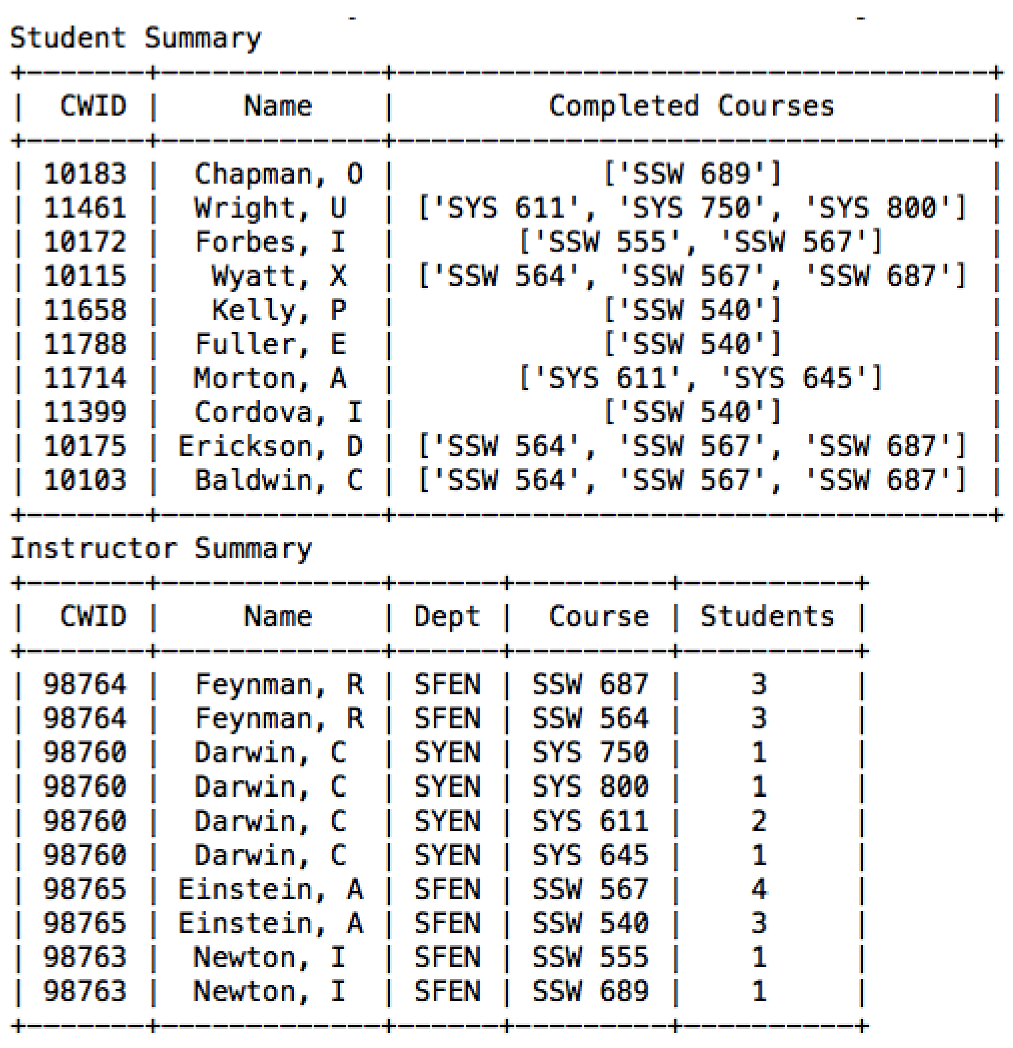
- Specifies the student CWID, course, and grade for that course, and the instructor CWID
- Each line has the format: Student CWID\tCourse\tLetterGrade\tInstructor CWID

Your assignment is to read the data from each of the three files and store it in a collection of classes that are easy to process to meet the following requirements:

- Your solution should allow a single program to create repositories for different sets of data files, e.g. one set of files for Stevens, another for Columbia University, and a third for NYU. Each university will have different directories of data files. Hint: you should define a class to hold all of the information for a specific university.
- You may hard code the names of each of the required data file names within a directory, e.g. "students.txt", "instructors.txt", and "grades.txt". Your solution should accept the directory path where the three data files exist so you can easily create multiple sets of input files for testing. I'll test your solution against multiple data directories representing different universities so be sure your solution allows me to easily run against multiple data directories.

- Use your file reader from HW08 to read the students, instructors, and grades files into appropriate data structures or classes.
  - **Print** warning messages if the input file doesn't exist or doesn't meet the expected format.
  - **Print** warning messages if the grades file includes an unknown instructor or student
- Use PrettyTable to generate and print a summary table of all of the students with their CWID, name, and a **sorted** list of the courses they've taken (as specified in the grades.txt file).
- Use PrettyTable to generate and print a summary table of each of the instructors who taught at least one course with their CWID, name, department, the course they've taught, and the number of students in each class.
- Implement automated tests to verify that the data in the prettytables matches the data from the input data files.
- You **do NOT** need to implement automated tests to catch all possible error conditions but your program should print relevant error messages when invalid or missing data is detected.
- Your solution SHOULD print error messages and warnings to the user about inconsistent or bad data, e.g. a data file with the wrong number of fields or a grade for an unknown student or instructor.

Here's an example of the **Student and Instructor summary tables. Note that the completed courses are sorted alphabetically**:



```
Student Summary
+-------+-------------+-----------------------------------+
|  CWID |     Name    |         Completed Courses         |
+-------+-------------+-----------------------------------+
| 10183 |  Chapman, O |            ['SSW 689']            |
| 11461 |  Wright, U  | ['SYS 611', 'SYS 750', 'SYS 800'] |
| 10172 |  Forbes, I  |       ['SSW 555', 'SSW 567']      |
| 10115 |   Wyatt, X  | ['SSW 564', 'SSW 567', 'SSW 687'] |
| 11658 |   Kelly, P  |            ['SSW 540']            |
| 11788 |  Fuller, E  |            ['SSW 540']            |
| 11714 |  Morton, A  |       ['SYS 611', 'SYS 645']      |
| 11399 |  Cordova, I |            ['SSW 540']            |
| 10175 | Erickson, D | ['SSW 564', 'SSW 567', 'SSW 687'] |
| 10103 |  Baldwin, C | ['SSW 564', 'SSW 567', 'SSW 687'] |
+-------+-------------+-----------------------------------+
Instructor Summary
+-------+-------------+------+---------+----------+
|  CWID |     Name    | Dept |  Course | Students |
+-------+-------------+------+---------+----------+
| 98764 |  Feynman, R | SFEN | SSW 687 |    3     |
| 98764 |  Feynman, R | SFEN | SSW 564 |    3     |
| 98760 |  Darwin, C  | SYEN | SYS 750 |    1     |
| 98760 |  Darwin, C  | SYEN | SYS 800 |    1     |
| 98760 |  Darwin, C  | SYEN | SYS 611 |    2     |
| 98760 |  Darwin, C  | SYEN | SYS 645 |    1     |
| 98765 | Einstein, A | SFEN | SSW 567 |    4     |
| 98765 | Einstein, A | SFEN | SSW 540 |    3     |
| 98763 |  Newton, I  | SFEN | SSW 555 |    1     |
| 98763 |  Newton, I  | SFEN | SSW 689 |    1     |
+-------+-------------+------+---------+----------+
```

Hints:

- Think through the overall design and sketch out the solution using CRC Cards (from lecture 2) with all of the major functionality before you write any code. My solution includes the following:
  - class **University** holds all of the students, instructors and grades for a single University. The class stores all of the data structures and methods together in a single place.
  - class **Student** holds all of the details of a student, including a dict to store the classes taken and the grade where the course is the key and the grade is the value.
  - class **Instructor** holds all of the details of an instructor, including a defaultdict(int) to store the names of the courses taught along with the number of students who have taken the course.
  - class **University** reads the grades file and either stores the data in a list, or just process each line as you read it from the file to update the courses taken by each student and to update the courses taught by each instructor and update the number of students
  - My solution does **not** define a class for grades. Instead, I store the information from the grades file in the Student and Instructor classes.
  - a main() routine to run the whole thing

- a test suite that compares known values for a directory of student, instructor, and grades files to the computed values.
- For each student, you need to know the student's name and major. We'll be adding other student attributes in the future, e.g. email address, etc. so a flexible solution will help long term.
- The key to your dictionary should uniquely identify the item, e.g. using the student's name as the key is **not** a good choice because two students may have the same name.
- We aren't using the grade for this assignment but future assignments may require us to consider if the student passed the course to determine if she needs to take the course again or to compute the student's overall GPA.
- Think about how you need to access the data when choosing the keys and values for each of the dictionaries. Recall that the value may be a dictionary, set, list, etc.
- Think about the operations that you need to perform on the data.

Test your program and upload your program to Canvas when ready. Be sure to handle unexpected cases, e.g. a student from the students.txt file who has no grades yet (she might be a first semester student). You can be sure that your testing group (Prof JR) will have some curious test cases to try against your solution.

We will continue to expand this task in the coming weeks so an elegant design is important.

How can you create automated tests for your solution? Hint: think about how we tested the numbers of lines, characters, functions, and class from HW08. You may want to create small student, instructor, and grades files to simplify your testing. Your automated tests should include tests that at least compare the information in the prettytables to the data from the data files.