Project 3: Student Class List

Problem

Write an application which stores the courses taken by a student and prints out a report. Data will be read from a data.txt file. Once all data has been read from the data.txt file, your program should print a report as shown in the Program Output section below. After the list of courses has been printed, you should then calculate and display the cumulative GPA of all courses in the list.

Important Constraints

1. You must implement a linked list as the underlying data structure.
2. Course entries will be stored in the linked list structure.
3. This must be done using classes and objects in OOP style. Each class will be in a separate module (see below) and the application code that reads the data from the data.txt file will be in file main.py.

Course ADT (course.py)

You will implement a Course ADT which implements the following methods:

- constructor: must have default values for all parameters and must validate all parameters.
- `number()`: retrieve Course Number as an integer
- `name()`: retrieve Course Name as a string
- `credit_hr()`: retrieve Credits as a floating-point number
- `grade()`: retrieve Grade as a numeric value in range 4.0 – 0.0
- `__str__()`: returns a string representing a single Course as shown in the Program Output section
- The courses taken by the student will be provided in a data file. You will have to parse the data from that file and store it in a Linked List of courses.

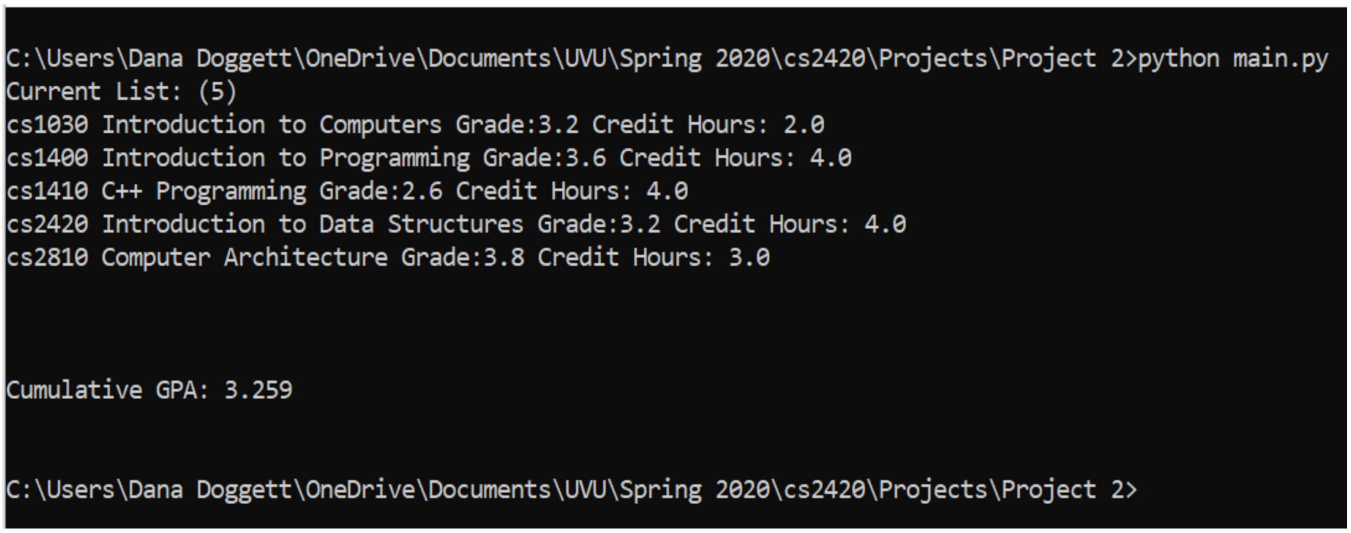
CourseList ADT (courselist.py)

Your CourseList ADT must use a Linked List implementation. The course list must be able to hold an **unlimited** number of Courses. The CourseList ADT implements the following methods:

- constructor to initialize all needed data for an empty list

- `insert(Course)`: insert the specified Course in Course Number ascending order
- `remove(number)`: remove the first occurrence of the specified Course
- `remove_all(number)`: removes ALL occurrences of the specified Course
- find(number): find the first occurrance of the specified course in the list or return -1
- `size()`: return the number of items in the list
- `calculate_gpa()`: return the GPA using all courses in the list
- `is_sorted()`: return True if the list is sorted by Course Number, False otherwise
- `__str__()`: returns a string with each Course's data on a separate line (as shown in the Program Output)
- `__iter__()` and `__next__()`: the linked list must be iterable

Program Output



```
C:\Users\Dana Doggett\OneDrive\Documents\UVU\Spring 2020\cs2420\Projects\Project 2>python main.py
Current List: (5)
cs1030 Introduction to Computers Grade:3.2 Credit Hours: 2.0
cs1400 Introduction to Programming Grade:3.6 Credit Hours: 4.0
cs1410 C++ Programming Grade:2.6 Credit Hours: 4.0
cs2420 Introduction to Data Structures Grade:3.2 Credit Hours: 4.0
cs2810 Computer Architecture Grade:3.8 Credit Hours: 3.0



Cumulative GPA: 3.259


C:\Users\Dana Doggett\OneDrive\Documents\UVU\Spring 2020\cs2420\Projects\Project 2>
```

Figure 1. Example Program Output

What to Submit in Canvas

- main.py (driver file)
- course.py
- courselist.py

Automated Tests

Your program will be run through a suite of automated tests to test your code for accuracy and functionality.

The following tests will be run:

- `test_course_creation()` – tests that course objects are created correctly
- `test_course_creation_with_parameters()` – verifies that course objects are created correctly when the constructor receives parameters
- `test_empty_courselist()` – ensures that an empty CourseList object has the expected properties
- `test_insert()` – tests that a course is added to a CourseList correctly
- `test_remove()` – verifies that a course object is removed from a CourseList correctly, and that the remaining list is correct
- `test_remove_all()` – ensures that a list is empty after all elements are removed
- `test_gpa()` – tests the gpa value calculated is accrurate for the contents of a given CourseList
- `test_iterate_list()` – verifies that a list can be iterated over
- `test_code_quality()` – ensures that your source code is written to the expected quality and style standards

These tests will be included with the project spec. Use `pytest` to verify your code works prior to submission.

Grading (100 points)

Pylint will be run on course.py and courselist.py. Expected minimum Pylint score for each file is 8.5.

Score is the sum of:

- Percentage of test cases passed x 80 points
- min(Coding style score/8.5, 1) x 20 points
- Possible adjustment due to physical inspection of the code, to make sure you implemented things.