CSE 131 Final Exam
Due: August 21st, at noon Central Time.

This is an open exam. You may consult any other sources that you wish, though you may not speak to other people about the content of this exam during the exam period. Any questions about the exam should be directed to the course instructor.

You can use code from your prior assignments. You can use resources that you find on the internet. The work that is submitted must be your own work, however. Submitting code from an online source or code that was written by anyone other than yourself will be considered a violation of the academic integrity policy and referred to the McKelvey School of Engineering for further action.

This exam will be graded based on the code that you commit and push to your code repository. Code that is dated after the end time of the exam your code repository will not be accepted, so make sure that you commit and push early and often.

Exam Mechanics:

You should create a repository by using this link:

All code for the exam should go in this repository. If your code is not in the repository by the end time of the exam, it will not be accepted. Please commit often, and verify that your work got committed by checking your repository page on GitHub.

If you need help setting up, please ask, but note that you will not receive extra time on the exam due to technical issues.

Exam Overview

Your goal for this exam is to create a simple gradebook application. This gradebook will be able to store the grades for all the assignments and exams for a given course. The gradebook should be able to retrieve and compute course grades for each student upon request.

Part 1 will have you complete a `Gradeable` interface that will be used for the graded components of a course.

Part 2 will have you implement the `Gradeable` interface on the two types of gradeable items: Exams and Assignments

Part 3 will have you implement the `Gradebook` functionality.

Also note that a main method has been provided to you in the `Gradebook` class. It contains examples of how to create and use the objects, so use this as a guide. You can and should add your own code to test your work more thoroughly: the provided tests are not comprehensive, and a more thorough set of tests will be run when your exam is graded.

Part 1: `Gradeable` interface (10 points)

An empty `Gradeable` interface has been created for you. Things that are gradeable should be able to:

- Return the total number of points that a `Gradeable` item is worth
- Return the total number of points that a student received for this `Gradeable` item
- Return whether this `Gradeable` item is an exam. If a `Gradeable` item is not an exam, you can assume that it is an assignment.

Part 2: `Exam` and `Assignment` (30 points)

An `Exam` **has-a**:

- Total number of points that the exam is worth
- Number of points earned by the student who took it
- A type (multiple choice, essay, etc.)

You should create a constructor for your `Exam` class. An `Exam` should implement the `Gradeable` interface. In addition to the functionality provided by the `Gradeable` interface, `Exam` should have a `toString`() method that displays the information it contains. Here is an example of what that might look like:

```
Multiple Choice exam: 91.0 / 100
```

An `Assignment` **has-a**:

- Total number of points that the assignment is worth
- Number of points earned by the student who submitted the assignment
- Name of the assignment
- Number of late days used on the assignment

You should create a constructor for your `Assignment` class. An `Assignment` should implement the `Gradeable` interface. In addition to the functionality provided by the `Gradeable` interface, `Assignment` should have a `toString`() method that displays the information it contains. Here is an example of what that might look like:

```
Assignment 1: 87.0 / 100, late days: 3
```

Part 3: `Gradebook`

A `Gradebook` **has-a**:

- Weight for the assignments. This should be expressed as a value between 0 and 1. For example, if assignments are worth 70% of the final grade, this would be represented as 0.7
- Weight for the exams, following similar rules above
- Name of the course
- A mapping of student names to a List of all the `Gradeable` items they have submitted for the course. This has already been defined for you in the `Gradebook` class and you must use it to store the graded items for each student in the course.

You can assume that the weights for exams and assignments will add up to be 1. You can also assume that the `Gradebook` will be empty when it is first created. You should create a constructor for your `Gradebook` class.

A Gradebook should be able to do the following:

- It should be able to take in a student's name and a `Gradeable` item, then add this information to the `Gradebook`
- It should be able to compute and return the grade for the exam portion of the course. This grade should be a percentage between 0 and 100. The percentage should be computed as the total number of points earned by the student on exams divided by the total number of points possible on exams.
- It should be able to compute and return the grade for the assignment portion of the course, following similar rules as above.
- It should be able to compute and return the current grade in the course for a given student. This means that the weights for exams and assignments should be applied to the respective exam and assignment grades.

- It should be able to print out the grades of all students who are currently in the gradebook. This output may look something like the following:

```
Grades for: Louis Cole
Assignment 1: 87.0 / 100, late days: 3
Multiple Choice exam: 91.0 / 100
Assignment 2: 63.0 / 80, late days: 0
Essay exam: 20.0 / 85
Exam Grade: 60.0
Assignment Grade: 83.33333333333334
Course Grade: 76.33333333333334

Grades for: Marc Rebillet
Assignment 1: 95.0 / 100, late days: 0
Assignment 2: 70.0 / 80, late days: 4
Multiple Choice exam: 70.0 / 100
Essay exam: 70.0 / 85
Exam Grade: 75.67567567567568
Assignment Grade: 91.66666666666666
Course Grade: 86.86936936936937
```

Again, please note that code examples of these methods being used as well as their expected output has been provided to you. Please use this to help you and add more tests of your own as appropriate.

Submission

To submit your assignment, simply commit and push your work. Please double check that it got submitted by visiting the repository page on GitHub. If you can see your work on GitHub then it has been properly submitted. Work that is committed after the due date will not be accepted.