

## Homework 3: Grading Yourself

Assigned:

Due:

*No extensions!*

**Summary:** In this assignment, you will explore the use of Scheme to compute class grades.

**Purposes:** To give you further experience writing expressions in “Scheme form” (prefix notation and parenthesized). To give you experience using files in Scheme. To encourage you to consider computations that might be automated.

**Expected Time:** Less than one hour.

**Collaboration:** You may work in any size group from one (i.e., alone) to four. You may discuss the assignment with anyone you wish. You may obtain help from anyone you wish, but you should clearly document that help.

**Submitting:** Email me your work. More details below.

**Warning:** So that this exercise is a learning assignment for everyone, I may spend class time publicly critiquing your work.

## Background: Grading

As you probably know by this point in your career, each faculty member has his or her own complex grading scheme. (That’s a lowercase scheme, as in plan or program of action, and not uppercase Scheme, which refers to the programming language.) One appropriate use of a programming language is to help you figure out your expected grade, given a grading scheme and a collection of grades.

The most straightforward technique is to write a formula (or formulae) for the grade and to fill in all the grades. To predict your final grade in the class, you fill in what you expect to get on subsequent assignments, based on what you’ve received on past assignments.

For example, suppose that the policy is that each of three examinations is worth 15%, a project is worth 20%, and the final is worth 35%. We might write

```
(define grade (+ (* exam1 .15)
                  (* exam2 .15)
                  (* exam3 .15)
                  (* project .20)
                  (* final .35)))
```

If, for some reason, a student had received an 80 on everything before the final, the student could consider possible grades based on the final.

```
(define exam1 80)
(define exam2 80)
(define exam3 80)
(define project 80)
```

If the student receives a 0 on the final, his or her class grade will be 52.

```
(define final 0)
(define grade ...)
---
> grade
52.0
```

If the student receives a 50 on the final, his or her class grade will be 69.5

```
(define final 0)
(define grade ...)
---
> grade
69.5
```

If the student receives a 90 on the final, his or her class grade will be 83.5, which might lead to a higher letter grade than he or she had before the final.

```
(define final 0)
(define grade ...)
---
> grade
69.5
```

We will consider a similar technique in this assignment.

(Just in case you care - more complex techniques can scale your current grades in estimating the final grade, or, given a desired final grade, can suggest what you need on the remaining work to earn that grade. We'll stick with the simpler method.)

## Assignment

Let us assume that we're working in a class in which there are five graded homework assignments, three graded examinations, a participation grade, a graded final examination, a class project, and an extra credit grade. We'll use the names `hw1 ... hw5`, `exam1 ... exam5`, `participation`, `final`, `project`, and `ec` for those grades.

a. Create a file, `sam-grades.scm`, that assigns the following numbers for those grades:

- Homework: 90, 80, 70, 70, 80
- Exams: 80, 85, 70
- Final: 85
- Project: 80
- Participation: 90
- EC: 5

- b. Create another file, `chris-grades.scm`, that assigns some set of grades of your choice.
- c. Create a file, `cs151-policy0.scm`, that computes a numeric grade, named `grade` using a simplified version of the policies of this class. Those simplified policies are as follows:

- Homework: 25%
- Participation: 15%
- Project: 15%
- Exams: 45%
- EC: added to the grade without scaling

Test your code by entering the following in a new definitions pane, and clicking Run. Try it for both sets of grades.

```
(load "sam-grades.scm")  
(load "cs151-policy0.scm")  
grade
```

- d. Create a file, `cs151-policy1.scm`, that computes a more sophisticated grade, using these additional policies:

- The lowest homework grade is dropped.
- The final examination replaces the lowest exam grade, if it is higher.
- Extra credit is capped at 3.

Test to make sure that this policy seems to compute a grade.

- e. Assume that Sam has completed all his/her work, except the final, and wants to figure out what grade he/she needs on the final to earn an A, B, or C. Using the techniques discussed above and the revised policy, determine those final score.

- f. Suppose that someone provided you with `cs151-policy1.scm`. What tests would you conduct to make sure that it works correctly? Note that you can assume that authors of the provided version were not trying to create a non-working version; rather, you are not confident that they have the technical skills to produce a correct version.

## Important Evaluation Criteria

The primary evaluation criterion for this assignment is, of course, correctness. That is, I will check to make sure that you expressed the specified grading scheme in Scheme.

Particularly elegant solutions may earn a grade boost. Conciseness is one aspect of elegance. Formatting of your code for clarity, using horizontal and vertical whitespace is another. You may discover others.

## Submitting Your Homework

Once you have ensured that all your work is correct, please submit your files as attachments to an email message.

In particular, open a mail composition window, either in Icedove or Outlook Express, and attach the three files described above. In the body of your message, answer parts e and f. Make the subject of the email *CSC151 Homework 3*. Send the mail.

## Hints

You should be able to do all of the computation with the following Scheme operations: `+`, `*`, `-`, and `min` (which computes the smallest of a group of numbers). You should find `min` particularly applicable to all three changes in part d.

---

Copyright © 2007 Janet Davis, Matthew Kluber, and Samuel A. Rebelsky. This material is based upon work partially supported by the National Science Foundation under Grant No. CCLI-0633090. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. This work is licensed under a Creative Commons Attribution-NonCommercial 2.5 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/2.5/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.