

COMP161

Project 1

Grade Explorer

Spring 2017

For this project you'll implement a basic, interactive CLI program that lets you carry out some standard tasks revolving around grades computed by weighted average.

Program Overview

Like many other courses, this course uses a weighted average system for computing your grade. Your assignments are broken down into categories where each category gets a weight expressed as a percent of the total grade. The final grade is then computed by multiplying the category grade by the weight and summing this value up for each category. A more detailed treatment of this is given below.

Your program will help students better understand their grade and this kind of system by carrying out the calculations needed to answer several standard questions:

1. What impact does a specific assignment have on the final grade?
2. What grade do I need to average to achieve a particular target grade?
3. What's my current grade?

The program should be interactive and takes no command-line arguments. When launched, the user will be presented with a menu that lets them select one of the above tasks or choose to quit the program. Each of the three core tasks are, themselves, repeated until the user chooses to return to the main menu. User inputs should be validated and appropriate prompts and error messages should be used. In short, you should design the UI with an eye towards usability.

Weighted Averages

Let's begin by looking at a basic unweighted average calculation¹ and see how it is a specific case of the weighted average. In this course your lab grade is the average of 10 lab grades. This means we add up each score and divide by the total number of labs. If l_0 is the score of the first lab and l_9 is the score of the last, then we compute the average lab score L as,

$$L = \frac{l_0 + l_1 + \cdots + l_8 + l_9}{10}$$

¹ the average you're most likely familiar with

Another way to write the sum of scores is with *Sigma* notation.

$$L = \frac{\sum_{i=0}^9 l_i}{10}$$

The greek letter \sum denotes a sum. To the right is an expression we wish to sum. Below the letter \sum is an initial value for a variable that will appear in the expression and above is the maximum value of that variable. So the expression given above is telling us: “For all values of i from 0 to 9, sum l_i ”². If we simply wanted to add all the numbers from 5 to 10000 we could write this as,

$$\sum_{i=5}^{10000} i$$

Returning now to the calculation of the average, we can express the average score A of n scores each denoted by s_0 to s_{n-1} as the sum,

$$A = \frac{\sum_{i=0}^{n-1} s_i}{n} = \sum_{i=0}^{n-1} \frac{s_i}{n} = \sum_{i=0}^{n-1} \frac{1}{n} s_i \quad (1)$$

The later two forms of Equation 1 come from distributing the multiplication of $1/n$ across the terms of the sum. Rather than first summing and then dividing, we can sum the scores after each has been multiplied by $1/n$ as each score accounts for $1/n$ of the total. The term $1/n$ can be viewed as the *weight* of the score with which it is multiplied. In this way the unweighted average is just a weighted average where every score has the same weight. This is often called an *unweighted average*.

A general weighted average allows for different weights so long as the sum of the weights is 1³. In the space of grades you see this expressed as a percentage of the total grade. To compute the weighted average A of n scores s_0 to s_{n-1} with respective weights w_0 to w_{n-1} we can compute the sum,

$$A = \sum_{i=0}^{n-1} w_i * s_i \quad (2)$$

If you check your syllabus you’ll see that your homework, lab, exam, and project grades are all determined by unweighted averages. The final grade is then a weighted average of these averages along with a participation grade. Each assignment category average is computed as a percent give the final weighted average a percentage point value as well.

² It’s useful to recognize that $\sum_{i=a}^b$ is a sum of $b - a + 1$ terms

³ the weight is the portion or percent of the whole

Task 1: Single Assignment Analysis

The first task in your program will allow the user to determine the impact of a single assignment on their grade. We'll need the assignment grade, the weight of that kind of assignment, and the total number of that kind of assignment. The program should then tell the user the weight of that assignment in their final grade and percentage points of their final grade gained and lost due to their score on that assignment. We'll now see how to work with parts of the weighted sum to compute these values.

Say your homework accounted for 30% of your grade and you had 6 equally weighted homework assignments, then what weight does a single homework have in the final grade? In the final grade calculation we'd take the total homework average H and multiply by the weight 0.3 to get the homework contribution to the final grade. Recall that as an unweighted average H would be computed as,

$$H = \sum_{i=0}^5 \frac{1}{6} s_i$$

where s_i is the score⁴ of the i^{th} homework assignment. Instead of multiplying H by the overall homework weight .3 we can distribute that weight across the individual terms of H to get the per assignment weight. In this case each assignment weight becomes $.3/6 = 0.05$ meaning a single homework assignment is worth 5% of your final grade. In general, if the assignment category has a weight of w_c and there are n assignments in that category, each weighted the same, then the per-assignment weight in the final grade is w_c/n .

⁴ presumably as a percent

Now that we know the weight of the assignment in the final grade calculation we can determine the number of percentage points it contributes to the final grade by simply multiplying the newly calculated weight by the actual score. For example, if we got 75% on a homework assignment, then by multiplying by 0.05, the weight of the assignment on the final grade, we get the contribution for this assignment towards the final grade. So this assignment contributes $0.75 * 0.05 = 0.0375$ or 3.75 of the 100 possible percentage points. On the other hand, we missed out on the other 25% of that assignment and have lost $0.25 * 0.05 = 0.0125$ or 1.25 percentage points. Put another way, as a result of this one assignment, our course grade can be no higher than 98.75% but no lower than 3.75%.

Task 2: Setting Goals

The first task of the program lets the user evaluate the present. The second task lets them think about the future. Given a current grade and the weight of that grade and a final goal, we can compute the

grade needed to reach that goal. Users should first provide the current grade and weight. They are then able to repeatedly enter a target grade in-order to see what their target average should be. The idea is that they can keep playing “What if?” with their grade, i.e. “What if I want to get a C?”, “Ok, and now if I want a B?”, etc. For each “what if” they should not have to re-enter the current grade and weight, only the goal.

This task works for any grade computed by an average. Let’s look at how the unweighted average used to compute assignment category grades becomes a weighted average in this context. Say you have nine labs in total and your average on the first five is 68%. The weight of those five labs is $\frac{5}{9}$ of your total lab score. The remaining labs account for the remaining $\frac{4}{9}$ of the lab score. Given an average of f on our remaining labs, we would compute the final lab score of L as follows,

$$L = \frac{5}{9}.68 + \frac{4}{9}f$$

With a bit of algebra we can re-write this as an equation for the average f .

$$f = \frac{L - \frac{5}{9}.68}{\frac{4}{9}}$$

What if we want a 80% on our labs? What do we need on our remaining four labs.

$$\begin{aligned} f &= \frac{0.8 - \frac{5}{9}.68}{\frac{4}{9}} \\ &= \frac{0.8 - 0.3\bar{7}}{0.\bar{4}} \\ &= 0.95 \end{aligned}$$

An average of 95% is needed on the remaining labs if we want to get an 80% for our lab grade.

In general terms, if we currently have an average of c , that average has a weight of w_c , and we wish to achieve a final grade of L , then the average t that we need to make or exceed on the remaining assignments is,

$$t = \frac{L - c * w_c}{(1 - w_c)}$$

This can be applied not only to assignment category grade but to the final grade as well. You’ll need to figure out the weight of the work done so far and the grade for that work, then the above formula can be used. The final task of the program gets at computing the grade and weight of work done for a specific category.

Task 3: Current Category Grade

The final task of your program computes the current score and weight on a particular assignment category. That is, given a set of assignment scores and the total number of assignments in that category, your program should report the current average for that category and the weight of that average in the final grade. When combined with the other functionalities of this program, users will be able to do some fairly detailed analysis of their grades.

The details of this calculation were implied by the second task. Let's do an example to make sure it's clear. If you have five exams total and on the first two you scored a 75% and a 88% respectively, then your current exam average is,

$$. \frac{.75}{2} * \frac{.88}{2} = 0.815$$

The weight in your final grade of this 81.5% exam average is $\frac{2}{5}$.

In general, if you've completed k of n assignments then the weight of those k assignments is $\frac{k}{n}$ and the average of those n is just their unweighted average.

Grading

Your grade for this project is determined by the quality of your code and the amount of the program completed correctly. The quality accounts for 40% of the grade and the completeness and correctness accounts for the remaining 60%. The completeness and correctness requirements direct you to produce a program that correctly carries out some of the desired features of this program. The quality requirements direct you to ensure that whatever you get done not only works but is done well to a standard beyond correct functionality.

Quality Points

To earn full credit for code quality you must use good programming style, have well written documentation, and a full set of tests for all procedures. Further more, your design should be making good use of helper procedures in order to manage the complexity of the program. Cramming everything into a small number of procedures is likely to cause you to lose some points on design quality but you can earn a good amount of points if the procedures you have are well documented and tested using gTest. On the other hand, you can do an excellent job breaking things down into basic procedures but lose points to sparse or missing documentation and tests.

Correctness and Completeness Points

This project is meant to be worked on incrementally in a top-down fashion. You should first build the complete UI with the core computational tasks stubbed out. You should then work through each task in the order they were presented. The UI and each task act as benchmarks for your completeness and correctness grade. Partial credit towards the next benchmark can be earned by having well done stubs and tests for procedures from that benchmark. In general, you should be incrementally developing the complete program where each increment is done well. Larger scale programs are not all or nothing endeavor. Having code that doesn't compile or that is ripe with serious bugs is not a goal of the programmer. It's better that your program do some of the work very well then all the work poorly.

An **A** level project will correctly carry out all three program tasks with little to no bugs.

A **B** level project will do everything the C level project can do but also be capable of carrying out the second task with little to no bugs.

A **C** level project will will do everything the C- level project does but also be capable of carrying out the first task with little to no bugs.

A **C-** level project will have a complete UI but none of the tasks are implemented. Users can select tasks from the main menu, and those tasks will go through their basic prompts, but only stub output is given. Stubs should exist for each task and should be called by the UI.

A **D** level project will compile and run but doesn't correctly navigate the UI nor do any of the tasks work. In short, it has serious, program-breaking bugs.

Any project that does not compile is unlikely to receive a passing grade.

Timeline

When it's due, submit your code as assignment *proj1* using handin.

<u>Date</u>	<u>Assignment Due</u>
3/29	Project Lab Assignment
4/5	(Nothing Due) Free Work time in Lab
4/15	Project due by "end of day"

Table 1: Project Due Dates