# How to Grade a Class that Has Both Theoretical and Practical Components?

Vignesh Ponraj and Vladik Kreinovich
Department of Computer Science, University of Texas at El Paso
500 W. University, El Paso, TX 79968, USA
vigneshponraj97@gmail.com, vladik@utep.edu

**How classes are usually graded: a reminder.** What do we want from students? For example, for CS1, we want them to do the practical stuff, to be able to code, and we want them to understand the theoretical part: how to trace the code, what for-loop does, and so on. The overall grade comes as a combination of these parts. How do you compute the overall grade?

You have a grade for the tests on which you basically test the student's ability to understand the code and to write down simple code. There is a grade for programming assignments, which test the ability of students to code some reasonable problems. Usually, the overall grade for the class is a weighted average of the practical and theoretical grades.

**Why is this a problem: a simple example.** Suppose that a student has a 100 on the theoretical part on all the tests, and on all programming assignments, only gets 60, below satisfactory. If we consider these grades equally, this gives us 80. So, not only the student passes the class, he/she is considered a good student.

Suppose now that upon graduation, this student is hired by a company. The company sees that on CS1, that student got a B. This means that the student is good, so he is given programming tasks. Unfortunately, on these tasks, he fails miserably. So, the company says: the university is inflating the grades. And they never hire our graduates again.

**A possible solution and its limitations.** A solution that some instructors use is to take the minimum of the two grades. So, if a student scored 100 for one part and 60 for another part, this student gets 60 and fails the class. A company looks at it and thinks that this student is useless in CS. However, in reality, for some tasks, the student can be useful. This student cannot write code, but he/she can understand others' codes and thus help.

A recent proposal. To resolve this situation, Dr. Nigel Ward from UTEP's Computer Science Department proposed the following empirical formula:

$$g = \min(\alpha \cdot g_p + (1-\alpha) \cdot g_t, \alpha \cdot g_t + (1-\alpha) \cdot g_p).$$

Here, $\alpha > 0.5$ is a constant. Dr. Ward uses this formula in his classes, and it seems to work.

But why this particular formula? What are the general kind of formulas that we can use? In this talk, we answer these questions.