211HomeworkGrading

How Scheme Homework Grades are Determined

The exact number of points available for each assignment (and each problem) will be determined based on their relative difficulty. The following instructions will be used in grading each problem.

Most homework problems will ask you to **develop a program** that solves a particular problem. The overall grade that you get for such a problem will be based:

- 25% on Correctness. Does the program pass all of our unit tests? If your program is correct, you will get all of these points. If not, you will get partial credit depending on how many of our unit tests your program passes. If you make a catastrophic mistake, you will probably receive zero correctness points.
- 25% on Testing. Does your program include a representative set of tests for each non-trivial program function, including non-trivial auxiliary functions? If so, you will full credit for testing. You need at least five tests for each function, including the base cases, examples of each form of data construction for each program input, and other inputs to make sure all of the legal edges in the program flow graph are covered.
- 25% on Design and Documentation. Is the overall design clean and intelligible? Are all of the requisite design and coding documentation present? This includes data definitions, templates, template instantiations, contracts, and purpose statements. The examples and tests for each function are already covered as part of testing.
- 25% on Program Style. Is your coding and presentation clean and easy-to-understand? This element of the grading rubric evaluates the quality of your program code. Both clean presentation (correct indentation, choosing good names for variables) and good low-level design choices (using reasonable auxiliary functions, avoiding code duplication, defining helpful constants) are important.

If your program returns an incorrect answer, please note this in your comments. (Do not elide that test case from your homework, and hope we don't notice!) It is better to state what bugs your function has, than to claim the code is correct when it's not; we will grade accordingly.

Some problems will ask students to **modify code already presented in the book**. In such cases, it doesn't make sense to follow the design recipe completely. However, tests (perhaps lifted out of the text as well) should be provided to insure that the new function has correct behavior. If the contract of the modified function has changed, the comments should reflect that. More generally, the usual problem-solving criteria will be used, but only as they apply to the part of the program that was modified. Other function definitions and comments lifted from the text can remain untouched.

Some problems ask students to "hand-evaluate" a program expression, reducing it to a value*. Follow the examples given in the Scheme HW Guide and include every step. Within an expression, you may elide subexpressions if there contents are obvious from conte3xt. Deductions will be made based on how far "off track" a reduction is.

Some problems require a **free response**. These can be provided in comments, and are expected to carefully answer the question. Grades will be determined by the correctness of your answer and the completeness of your reasoning.

Many problems ask multiple questions or split the problem into **multiple subproblems**. In this case, each subproblem is allocated a certain number of points and graded separately. It's important to answer *all* the questions that a problem asks, not just the first one.