Program Grading

Your Programs will be graded on two major areas: functionality and design/style. Although you will often concentrate your efforts on getting the program "to work" (whatever that takes!) you'll learn that our expectations go beyond just working— we also want an elegant, well-engineered solution. A program can truly be a work of art and something you can be proud of when showing off both the executable and the code.

Functionality Functionality denotes whether or not the program performed as expected i.e. did it meet all the requirements set out in the assignment handout. For this evaluation, we examine your program's behavior from an external perspective, without looking at the code. We will test your program on many different inputs to determine its correctness in a variety of situations.

Design/style Design and style capture whether the program has a clean and elegant organization that is easy to read and understand. Does it employ principles of good decomposition and re-use/unify rather than repeat code? Does it make well-reasoned choices for data structures? Would it be easy to maintain and/or extend? Are the variables and procedures meaningfully named? Does it have descriptive and accurate comments? Are the layout choices readable and consistent (use of white space, capitalization, punctuation, etc.)? For this evaluation, we carefully read the code and offer constructive input on your choices.

Each category is assigned a grade using a bucket scale (the two different scales used are described below). Rather than resorting to nit-picking and summing up points, we want to provide a clear overall sense of your program's strengths and weaknesses in these areas, and we believe a fairly coarse bucket scale is best for this. The total assignment score is the sum of the scores, the two parts are equally weighted. It is possible for a perfectly functional program to earn a low score in style and vice versa. We encourage you to make your program shine in both areas!

Interactive grading

Programs will be graded in a one-on-one conference with your section leader. Past 106 students seem in strong agreement that this interactive grading is one of the course's best features. Your section leader will discuss with you the specifics of where your program excels and where you can improve for next time. Their feedback and nourishment can help guide you in your passage from programming novice to accomplished software engineer. The grading conference is a marvelous and rare opportunity for intense, personal feedback. I hope you'll find it a lot more useful and inspiring that getting a returned paper marked with a numeric tally.

Functionality scale

Below is an explanation of the scale we will be using to express the level of functionality in a given program. Pay special attention to the **message**— each bucket is designed to send a clear indication of where you stand relative to our standards that should help you decide how to proceed from here.

+ An exceptional program—one that works correctly in all situations and exceeds our expectations, going above and beyond the assignment requirements. Super work! We cannot find fault and would love to show such a program to the entire class as an extraordinary example. Only about 5% of the programs receive this grade and it is never given out unless a ✓+ was also received for style. The message: "Fantastic work, congratulations!"

✓+ An excellent program—one that meets all of the requirements gracefully and has only minor flaws. Your section leader may be able to provide suggestions for some small adjustments or quibble with a few things, but these programs are overall very well-done and distinguish themselves in all areas. Perhaps a third of the programs will receive this grade. The message: "Great job, keep it up!"

✓ A good, solid program—one that is complete and generally meets the requirements of the assignment, although perhaps some parts could have been handled better. Your section leader can provide feedback on what can be improved upon for next time. This is the median grade and we expect around half of the programs will be given this grade. The message: "Good work."

✓- A program with room for improvement. The program doesn't quite meet all of the requirements of the assignment, perhaps because it is a bit incomplete or has errors with one or more required components. With a bit more work it could be a good program. Your section leader can help you to understand where to go from here. The message: "Hang in there, try to learn what you need to move forward."

- A troubled program. The program exhibits serious problems and/or is an incomplete attempt, but it does show some understanding and effort. This grade indicates that improvement needs to be made immediately. The message: "Time to get serious about getting back on track."

-- A very incomplete program. This score usually indicates the program had very little effort put into it and/or has serious errors. A large amount of work needs to be done before this program would work correctly. The message: "Talk to instructor before it's too late!"

0 No program turned in at all.

On some assignments there will be an opportunity for extensions beyond the standard assignment requirements. Doing this extra credit will potentially earn you between one and four Gold Stars of Valour. These will be figured into your final grade, but not your assignment grade. Also, note that while they do count, extra credit will not make up for a problematic assignment. Thus make sure you have thoroughly tested your assignment core before moving on to any extensions! Ultimately the main part of the assignment is the real meat and extensions are just gravy.

Design/style scale

There are fewer buckets in the design/style scale and we hope for the scores to be a bit higher than functionality. Below are the buckets used for this evaluation:

√+	A well-engineered, sensible, and readable program. Bravo! The overall approach is straight-forward, data structure is cleanly organized, tasks are nicely decomposed, algorithms are clear and easy to follow, comments are helpful, layout is consistent— an overall polished production. All students are capable of achieving this result and we encourage you to set your goal high. We expect it will take some feedback and iteration before your work is this graceful, but by the end of the quarter we would love to give out half or more grades in this bucket.
√	A program that exhibits reasonably good design and style. The program clearly makes a sincere effort toward an elegant solution, but it may not be entirely consistent, have a few minor problems or perhaps a more major issue that needs attention. Your section leader will be able to offer constructive ideas on how to tidy up and refine your work.
√-	A problematic program. The program has more significant flaws in terms of design and style, either due to carelessness or lack of coaching on these important aspects of programming. We encourage you to work closely with your section leader to develop better awareness of these issues and are happy to offer lots of concrete suggestions for improvement. Certain programs, such as those that are very incomplete, will automatically be placed in this bucket.
0	No program turned in at all.

A typical criteria

Although your particular section leader will personally be responsible for evaluating your program and meeting with you to discuss the feedback, all section leaders are working from standardized criteria that establish our class-wide grading standards. The course staff collaborates on drawing up specific criteria for each assignment and then each section leader applies the same criteria to their students' submissions.

Our functionality criteria typically consist of a listing of specific test cases to exercise the program on. The section leader compiles and runs the student program, attempts each of the specified test cases, and summarizes the results. We design our test cases to be as isolated as possible so that a previous error doesn't cascade into another. Each test case is weighted to account for its relative importance to the overall substance of the program. Significant weighting is given to the mainstream features (i.e. the most common inputs, straightforward cases) and less attention is paid to the more fringe behavior (i.e. error handling, edge cases). For a program such as Battleship, some of the mainstream test cases would be observing the program's handling of user input (clicking on an empty cell, hitting a ship, sinking a ship, etc.). Clicking off the board is an example of a more minor concern. Severe functionality errors include missing required features and any sort of crash under normal operation.

To earn a functionality ✓+, a program should not exhibit any major functionality flaws. It might have an issue or two some minor functional details. A ✓ program may have a bit more troubles, but will still largely work as required by the assignment specification. Programs that earn ✓- and below typically represent programs with more serious functionality issues— unimplemented/broken features, program crashes, and the like.

Our design/style criteria would be a specific list of issues to consider for the particular assignment in question. Like the functionality entries, the issues are weighted to account for their relative importance. An example of a major design/style error would be a poorly designed data structure that necessitates a lot of awkward coding through the program. A medium-importance issue might be failing to seize an opportunity for code unification or factoring to avoid code repetition. More minor issues would be forgetting to use named constants or inconsistent capitalization. Some issues, such as inadequate commenting or poor decomposition, can be very major (if the entire program suffers from it) to quite minor (if only a small portion is affected).

In order to earn a ✓+, a program should have no major designs/style flaws, although we may have some suggestions for improvement on a few smaller issues. Overall these programs are examples we would be happy to share with other students to demonstrate what to strive for. A ✓ program represents a solid attempt, but with some medium/minor missteps (or perhaps even an isolated major issue) and we know you can do better with some good feedback and a little more focused effort. ✓- represent programs with more significant concerns. With some directed coaching on the particular issues at hand, willingness to ask for advice when in doubt, and prioritizing these issues, most students are able to bounce back from any early design mishaps,

Lateness

After your self-granted extensions have been exhausted, late programs are penalized approximately one bucket per class day late. Furthermore, note that no assignment will be accepted more than three class days after its original due date. This is a strong incentive for you to plan your schedule to meet the deadlines and conserve your valuable late days for true necessity. Since the programs are assigned one after another, it is also important to consider the impact of late work on the next assignment. Don't let the "domino effect" bring your entire quarter down!