Assessment policies

Mark Armstrong

August 11, 2021

Contents

1 Automated unit testing policy

Automated unit tests will be provided for all assignments, and whenever possible and practical, they will also be provided for each homework.

You should not submit the testing files with your assignment contents; it will not cause any problems, but testing files will be ignored and overwritten before testing.

Alongside the files for unit testing, a Docker image will also be provided, in order to ensure that you are able to run the tests in the exact same environment that the course staff will use.

Passing the provided tests is *mandatory*, but **does not** *guarantee* a passing grade (both for homeworks and assignments.)

- Assignments will undergo a code review by the course staff, and your grade will be influenced just as much or more by your code's *approach* and *style* as by passing of tests.
- Homeworks will undergo a similar, but much more cursory, code review. Barring any obvious issues, you should receive a passing grade if your code passes the tests.

Submissions which do not pass all or a majority of the tests **may not be considered for grading at all**, at the discretion of the course staff.

During marking, we will typically add some *additional tests*, often constructed to test what we consider to be more "extreme" cases than are covered by the the provided tests, possibly including interesting edge cases.

- You are encouraged to try and think of these cases yourselves, and add appropriate tests to the provided ones in order to better check your solutions.
- You are not expected to submit any updates or additions to the testing files; as mentioned, any submissions of testing files will be ignored and overwritten.

2 Assignment literate documentation

2.1 Assignment literate documentation policy

In addition to source code files, assignments will also require you submit *documentation* for your code in the form of a literate programming document.

20% of each assignment's marks are set aside for this documentation.

- 12% for the contents of the documentation, and
- 8% for the style of the documentation.
- Even if the assignment is incomplete, full documentation marks may be awarded,
  - so long as some parts are sufficiently completed,
  - and some discussion of the difficulties with missing parts is included.
    - (More than just "I ran out of time".)

Any of the following formats are acceptable for this documentation:

- Markdown (homepage)
- Org mode (homepage)
  - Implementations outside of Emacs exist, but typically have somewhat minimal features.

- ReStructured text (homepage)
- HTML
  - Using `<code>` tags,
  - and preferably a tool to provide syntax highlighting, such as
    * highlight.js or
    * Prism
- PDF
  - In particular, through LaTeX (homepage)
    * Using a package such as `listings` (documentation) or `minted`, which provides syntax highlighting (GitHub homepage and documentation)
- Possibly more; speak to us if there is a format you feel should be accepted.
  - Microsoft Word, OpenOffice and other WSIWYG (What You See Is What You Get) editor formats will not be accepted.
    * If you wish to use such an editor, you may export your file to a PDF for submission. Do be sure to follow the style guidelines below.

2.2 Assignment literate documentation style guide

As mentioned in the assignment literate documentation policy, 8% of the marks of each assignment is allocated for the style of the documentation.

The section general literate documenation style rules below outlines what is required in your documentation, and what is optional.

Then the section format specific literate documentation style rules are some comments about style requirements or recommendations for specific formats.

2.2.1 General literate documentation style rules

Required:

- In non-plaintext formats (such as HTML and PDF), code blocks **must** be displayed using fixed width (monospace) fonts. (What's a fixed width font?)

  - In *non*-plaintext formats (such as HTML and PDF), non-code-block portions should be displayed using non-fixed width fonts.
  - There *must* be a font distinction between

- Headings (and often subheadings) must be used for organisation.
  - Typically, it is sufficient to use the same document structure as in the assignment description.
- Code blocks must not be too long; there should be documentation interspersed with code regularly.
  - There is not a hard and fast rule here; instead follow these guidelines:
    * Where they are more than a few lines, each function, procedure, type declaration, etc., should be in its own code block.
    * In no instance should a code block span an entire page or more.

2.2.2 Format specific literate documentation style rules

:TODO:

2.3 Assignment literate documentation content guide

:TODO: