Contents

Overview

There are three different parts, which are evaluated with slightly different emphases:

- Part A: Brightness
- Part B: Readaline
- Part C: Simlines

The distribution for FC, S&O and design document for this assignment are weighted as follows:

Design	10%
Brightness SO	5%
Brightness FC	15%
Readaline SO	10%
Readaline FC	10%
Simlines SO	25%
Simlines FC	25%

1 Part 1: Brightness

- Do the functions provided have a clear contract, either with descriptive names or with clear and concise written documentation? If written documentation was provided, was it free of a narrative description of the code?
- Was the code written in a modular way by providing concise compositional functions? In particular, was there a separation between computing the average brightness and printing the average brightness to `stdout`?
- Was the return value of `fopen` checked to ensure successful open?
- General adherence to course coding standards.

1.1 Grades

Very Good	Mostly good documentation and modular code with minor errors
Good	Modular code with unclear documentation or mostly clear documentation with not modular code
Fair	Unclear documentation and not modular code
Poor	No documentation and is very hard to follow

2 Part 2: Readaline

- Was the code clear, in terms of using good function and variable names, and in terms of clear documentation and explicit function contracts?
- Was the return value of `malloc` / `calloc` / `realloc` checked to ensure there is no memory error that may result in segmentation fault?
- General adherence to course coding standards.

2.1 Grades

Very Good	Mostly good documentation and modular code with minor errors
Good	Modular code with unclear documentation or mostly clear documentation with not modular code
Fair	Unclear documentation and not modular code
Poor	No documentation and is very hard to follow

3 Part 3: Simlines

- Was the code clear, in terms of using good function and variable names, and in terms of clear documentation and explicit function contracts?
- Was the function composition appropriate and were the choices made for dividing the problem into different functions appropriate?
- Was the return value of `malloc` / `calloc` / `realloc` checked to ensure there is no memory error that may result in segmentation fault?
- General adherence to course coding standards.

3.1 Grades

Very Good	Mostly good documentation and modular code with minor errors in formatting
Good	Modular code with unclear documentation or mostly clear documentation with not modular code
Fair	Unclear documentation and not modular code
Poor	No documentation and is very hard to follow or no modularity at all (entire code in main)

As this is the first assignment, we want to make sure that they are following the formatting guidelines. We need to point out all their errors, and if a student has a lot of formatting errors (like not following most of the guidelines) we reduce the grade by one level.