We normally publish the post-mortem for an assignment after it has been marked and released. Here is a list of common errors provided by the graders for assignment 2.

**Style and Spacing**

- Constants (and helper functions) should be defined above the design recipe for the function they are used in. Many students defined their constants and helper functions between their examples and function definition, or after the main functions specified in the assignment.

- Helper functions require purpose, contract, and one to three examples.

- Lines should be less than 80 characters long: excessively long lines should be broken up into multiple shorter lines, and make use of DrRacket’s auto-indenting features (Ctrl + i or Command + i).

- Some students forgot to leave a blank line before and after function definitions, or included both the function header and part of the function body on the same line.

**General**

- Remember to check basic test results after submitting to catch simple (but impactful) errors.

- Many students used parameter names that were not meaningful (e.g., a, b, c) or used ambiguous/unclear names (e.g., lp, we, jp). It is better to have names that are long and clear than names that are short and cryptic.

- The names of helper functions should also be clear and descriptive. For example, helper, test, input, function, and compute are very poor choices for helper function names.

- For now, equal? should only be used to compare two values of unknown types, or values that can take on more than one type. When the types of the arguments are known, use the most appropriate comparison function (e.g., symbol=?).

- To determine if a boolean value is true or false, use that value directly. Expressions like (boolean=? param true) or (equal? param false) are overly complex.

- Starting a new cond expression in an else clause is unneeded. Instead, directly check for the next condition in the original cond expression.
Design Recipe

- Purposes should begin with a function header (e.g., `(func-name param1 param2 param3)`). These headers should include each of the parameter names used in the function.

- Purposes should meaningfully use each parameter name in the description of the function, and these references to the parameter names should be written exactly as they appear in the function header.

- Contracts should begin with `func-name: `.

- Requirement is needed for symbols to specify what kind of symbol the function can take.

- If restrictions are already implied in a data type, they do not need to be included in the `requires:` section.

- Requirements are not needed for the values and types that functions produce.

Question 1

- Many students did not have a condition clause in their function for the case that number of lines is bigger than 4.

- Many students did not define constants for base points for each line.

Question 2

- Many students did not define constants for the age brackets or did not define constants for points associated with age.

- In part (a), many students did not make use of any helper functions, and had a large block of code for their solution in `pr-cec-score`. Although helper functions do not decrease code duplication in this question, they should be used for clarity, and to break down the problem into smaller parts.

- Some students did not make use of patterns when computing the points associated with each age, and enumerated all the possibilities from the ages 30 to 49 in their solution.

- In part (b), many students did not use `pr-cec-score` as a helper function, which lead to unnecessary code duplication.

- Many students did not include a test case for `pr-cec-eligible?` where the value produced from calling `pr-cec-score` would be exactly 350. This test should be included, as 350 is considered a break point in the context of this question.

- For both parts (a), and (b), many students did not include contract requirements for language proficiency, and the education level. As a result, some additional students had test cases with invalid input, e.g. language level 0.

Question 3

- In part (a), many students used `and`, `or`, `not` as part of their condition clause, such that their solution failed all correctness tests.

- Many students did not define constants for number-of-days thresholds of 2, 4 and 10.
• Purposes should only describe what the function does and not how it does it (i.e., they should not specify that the function was implemented using only cond/boolean expressions).

• Some students copied their test cases from (a) into (b) but forgot to change the function called in their check-expect.