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.

General

- Many students did not reference parameters, by name, in their purpose statements.
- Some students specified requirements for their functions in their purpose statements instead of the `requires` section of the contract.
- Some students used type names in the `requires` section of the contract instead of parameter names (e.g. `Num > 0`). This can cause ambiguity over which parameter the requirements apply to.
- Some students had contract formatting issues outside of the `requires` section. Common errors include:
  - Placing the function name in brackets. `(fn-name): Nat -> Nat`
  - Including parameters with the function name. `(fn-name param): Nat -> Nat`
  - Using parameter names instead of type names. `fn-name: param -> Nat`

The contract is meant to give you a clear understanding of the types a function consumes and produces. For this reason, consistent and accurate style is expected, while parameter names are excluded.

- Some students had a helper function or constant interrupt the design recipe of a main function.
- There were a variety of formatting issues with `(listof X)`. Common issues include missing brackets (i.e. `listof X`) and incorrect capitalization (e.g. `(ListOf X), (listOf X)`).

Question 2 (extremes)

- Some students missed the non-empty list requirement.
- In terms of correctness, this question was well done.

Question 3 (grades)

- Many students did not define constants for the weights associated with each grade component. It is extremely important to avoid "magic numbers" in your function body for the sake of both readability and adaptability.
- Many students also did not define a constant for the failing grade of 46.
- In terms of correctness, this question was well done.
Question 4 (pos)

- Many students missed the requirement that each element in costs has to be greater than or equal to zero in all parts of this question.
- Many students missed the requirement that there exists a number in costs that, if reduced to zero, would mean paid $\geq$ the sum of the numbers in costs in part (c). This was specified in the assignment and is crucial to the function’s implementation.

Question 5 (parity)

- This question was well done overall.
- Some students missed the requirement that the input binary string should only contain 1 and 0.

Question 6 (mean-relative)

- There were some instances of [else (cond ...)]. This is very poor style: if cond is used after else, it means there should have been another condition specified before the else.

Question 7 (sudoku)

- Some students had repeated code that could have been eliminated by a helper function (marks were not deducted).

Spacing

- Lines should be less than 80 characters long: excessively long lines should be broken up into multiple shorter lines.
- Note that DrRacket has auto-indenting features. You may use Ctrl-I or cmd-I for auto-indentation. Be careful when using auto-indentation: the 80 character wide restriction takes priority. Students are encouraged to review section 4.4 of the Style Guide.
- Many students did not have consistent spacing. Students are encouraged to review the design recipe’s specifications for proper spacing and indentation.