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.
- Helper functions require their own purpose, contract, and examples.
- Lines should be less than 80 characters long: excessively long lines should be broken up into multiple shorter lines.
- Some students did not use DrRacket’s auto-indenting features and had inconsistent indenting.
- Some students forgot to leave a blank line before and after function definitions or two blank lines before and after function blocks (design recipes).
- Some students used inconsistent spacing before/after operators/brackets, eg. (+(/ 2 3)4) instead of (+ (/ 2 3) 4).

**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., x) or used ambiguous/unclear names (e.g., fe, m1).
- equal? should only be used to compare two values of unknown 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) 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.
- Symbol values should not be defined as constants. This does not provide any additional meaning.
Design Recipe

- Purposes, contracts, and function definitions should not be explicitly labelled.
- 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.
- Contracts should begin with “func-name:”.
- Requirements are not needed for the output of functions.
- `Bool` is defined as being true or false. A requirement is not needed for this.

Question 1

- Some students included redundant cases in their `cond` statements. Don’t retest conditions that have already been proven to be false.

Question 2

- Many students did not define constants for the AirMile rates.
- Many students did not include requirements for the valid card types or for valid purchase amounts.
- Some students did not correctly specify that the function produces a `Nat`.

Question 3

- Many students used `Num` in their contracts instead of `Nat` or `Int`.
- Some students did not require the consumed grades to be between 0 and 100 inclusive.
- Many students did not define constants for mark weights or for the special values 46 and 50.

Question 4

- Many students did not include requirements for the valid blood types.
- There were three logical simplifications that could be made for valid blood type pairs (the universal donor, the universal recipient, and donating to the same recipient type). Many students did not include one or more of these simplifications.
- 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/bool` expressions).
- Some students did not include proper design recipe for Q4b and instead asked the reader to refer to Q4a’s design recipe. Each function’s design recipe should independent of other design recipes.