Post-Mortem
Assignment 04
October 17, 2019

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 4.

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. Students are advised to refer to pages 18-19 of the style guide for a sample submission that includes constant and helper function definitions.

• 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.

• If the `cond` question/answer pair does not fit in one line and causes the code to look too horizontal, students are advised to put the answer component on a new line.

General

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

• Some students code did not run at all, hence they lost all correctness marks. You are advised to make sure your code runs.

• Some students still had black highlighting present in their code, which is a clear indication of a lack of thorough testing. Students are advised to - as an absolute minimum - ensure there is no black highlighting in their code once it has been run.

• The names of helper functions should also be clear and descriptive. For example, `helper-loc`, `function`, and `p` are very poor choices for helper function names.

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, and these references to the parameter names should be written exactly as they appear in the function header.
• If restrictions are already implied in a data type, they do not need to be included in the `requires:` section. For example, if a function consumes a `Nat`, there is no need to specify that the consumed `Nat` should be greater than or equal to 0.

• Some students forgot to capitalize the type names in their contract (e.g. using `nat` instead of `Nat`).

Question 1

• Some students did not use a helper function to deal with a list of characters, and they did not make use of the built-in Racket function `substring` as well, which resulted in messy code with a lot of conversion between lists and strings using `string->list` and `list->string`. This approach is highly discouraged. Whenever appropriate, please use helper functions that deal with a list of characters and make your main function a wrapper for it.

Question 2

• This question was very well done!

Question 3

• Many students did not define constants for the values of a nickel, dime, quarter, loonie and toonie. Using constants in this case greatly helps with code readability and reduces the chances of making simple errors. For example, a statement like `(cond [(symbol=? coin 'nickel) 10])` is a harder mistake to catch than `(cond [(symbol=? coin 'nickel) dime-value])`.

• Some students defined constants for symbol values (e.g. `(define loonie 'loonie)`). Symbols are self-documenting and thus do not need to have a constant defined for them.

• In part (a), many students had contracts stating that the result of the function was a `Num` when it was a `Nat`. Please use more specific types in your contract whenever applicable.

• Some students only had one example for both of the main functions for this question - at least two examples are expected here.

Question 4

• Many students used the `Num` type in their contracts (without any requirements) even though the question specified that the functions consume `Nats`.

• Some students had overly-complex solutions that resulted in them timing-out on some correctness tests. Please refer to the sample solutions for a cleaner and faster solution.

• Overall, this question was well done!