Post-Mortem

Assignment 03

February 11, 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 3.

Style and Spacing

- Parameter, function, and constant names should generally be lower case, include dashes in between words, and not use any underscores.
- Constants (and helper functions) should be defined **above** the design recipe for the function they are used in. Some students are still defining their constants and helper functions between their examples and function definition, or after the main functions specified in the assignment.

General

- Students are reminded to only submit plain-text files to MarkUs.
- Many students did not make use of helper functions to reduce large chunks of repetitive code throughout the assignment.
- Predicates should not be written in the form `(cond [pred? true] [else false])` or `(cond [pred? false] [else true])`, where `pred?` is any boolean expression. This is unnecessarily complex, as the function body can be simplified to include only the boolean expression `pred?` or `(not pred?)` itself.
- 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=?`).
- Starting a new `cond` expression immediately in an `else` clause is unneeded. Instead, directly check for the next condition in the original `cond` expression.
- Some students submitted code that did not run and thus lost all their correctness marks. Students are highly encouraged to make sure their code runs.

Design Recipe

- Types in contracts should **always** be capitalized. This includes any user-defined structures like a `Clicker` or `CS135Mark`.
- Helper functions require their own purpose, contract, and examples.

- Any requirements that are present in a data definition do not have to be repeated in a contract for a function that consumes that type.
- 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.
- Requirements specified in the assignment question should be mentioned in the design recipe.

Question 2

- Some students did not write a helper function to find the distance between two `Posns`, which was implicitly required for a clean implementation.
- Some students defined three functions that calculate the distance between two `Posns`. This indicates a lack of understanding of how functions work and is very bad style.
- Students are reminded to use the tolerance specified in the assignment in their `check-expect` tests.

Question 3

- Many students did not make use of helper functions in their solution which resulted in extremely complex code.
- Many students did not use constants for their solutions in Q3a and Q3b, indicating bad style.
- Many students forgot to include the requirements described in the question in the requires section of their contract.
- A few students did not use the `CS135Mark` structure in their solution for Q3b and Q3c, which defeats the purpose of the question.
- Students were expected to use their solution for Q3a in Q3b and their solution for Q3b in Q3c. Defining a separate helper with the same body as a previously defined function or copying the body of a previously defined function in the body of another function duplicates code and indicates extremely poor style.

Question 4

- Some students misread the question and incorrectly translated the equations. Please remember to read the qustion carefully.
- Many students missed stating the requirements that causes division by zero.
- Overall, this question was done well.