Post-Mortem

Assignment 03

February 6, 2018

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 still defining their constants and helper functions between their examples and function definition, or after the main functions specified in the assignment.

General

- As specified in the assignment preamble, any symbol values should exactly match the description in the questions. Many students used symbols that differed slightly from those described in the assignment for their check-expects (such as using ‘stripe instead of ‘striped in question 3, or using ‘diamond instead of ‘diamonds in question 4). Since using these symbols would not exactly satisfy the data definition for a SetCard or a Card, they were not considered valid tests, and thus could not be picked up automatically. Please make sure that all your check-expects are valid tests on future assignments.
- 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=?).
- 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 immediately 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. As stated in the course notes, they are self-documenting, and defining them as constants does not provide any additional meaning.
Design Recipe

- Types in contracts should **always** be capitalized. This includes any user-defined structures like a FileInfo or SetCard.
- 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.

Question 2

- In part (a), many students did not include the requirement that the consumed string must contain only alphanumeric characters, if it is non-empty.
- In part (a), some students misinterpreted the question, and either always appended “new” to the end of the path, or always appended “copy” to the end of the name.
- In part (b), many students did not define constants for the compression rates of 40%, 25%, and 50%.

Question 3

- Many students either did not make use of helper functions in their solution, or had helper functions which were nearly identical.
  - In part (b), many students had three helper functions to deal with colours, shapes, and shadings separately. By selecting out the individual fields at the instance that the helper functions are called, instead of calling the helper functions with the entire SetCard structures, this can be condensed into one helper function.
  - In part (c), many students enumerated all the possible combinations for colours, shapes, and shadings with three separate helper functions. By including parameters that keep track of all the possible options for colours, shapes, or shadings, this can also be condensed into one helper function.
- In parts (b) and (c), many students did not include the requirements that the consumed SetCards must be different.

Question 4

- Many students did not define constants for the rank value of an Ace (11), or the rank value of a Jack, Queen, and King (10).
- Some students missed the fact that an Ace, Jack, Queen, and King have a different rank value than their face value, which lead to a loss of correctness marks throughout the question.
- In part (a), some students did not define helper functions for converting a card’s face value to its rank value, or for checking whether the consumed symbol matched any of the cards in the consumed Hand.
• In parts (b) and (c), some students did not use `points-by-suit` or `points` as helper functions respectively, which often lead to large chunks of repetitive code.

• In part (c), many students did not correctly account for the case where the card picked has a higher rank value than a card in the consumed `Hand`, that could be replaced without changing the points value of the `Hand`. 