Post-Mortem

Assignment 05

October 28, 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 5.

Style and Spacing

• Data definitions should not interrupt a function’s design recipe. For example, the data definition of a Book should not be written in between the purpose and the contract of a function. A good place to put your data definitions would be at the top of your file, or before the function you’re using the data definition for.

• 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.

General

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

• Helper functions should be defined before the functions that use them, and their definition should not interrupt the main function’s design recipe.

Design Recipe

• 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).

• Some students capitalized false in their contract - as per the design recipe, type names should be capitalized, but specific values such as false, "Sharon", 'alaska should be written exactly as they are.

• Template functions require a contract, and their produced type should be Any. For example, a template for a type called ‘Yeet’ should have the contract ;; yeet-template: Yeet -> Any.
Question 2

- In part (a), many students wrote Nat and (listof Nat) instead of Num and (listof Num). Please pay close attention to the directions of the question.
- In part (a), many students did not include false as a produced value in their contract.
- In part (b), some students wrote (listof (listof Num Str)) as the produced type. (listof X) should only have one type associated with it. (listof (listof (anyof Num Str))) is acceptable, even though (listof (list Num Str)) is preferred since it conveys more information about the produced value - i.e. each element in the produced list is a list of length 2, with a Num as the first element and a Str as the second.
- In part (b), some students had only included one example when at least two were warranted.
- In this question, some students had incorrectly formatted the (listof X) type. Some example of incorrect formatting are Listof X, list of X, List, and (listof) X.

Question 3

- Some students had a recursive solution for book-template. Since we know that a book is a list of exactly two elements, the template does not need to use recursion.
- Some students used cond and empty? statements in book-template, which is unnecessary.
- For listof-book-template, many students did not call book-template on (first list-of-books). However, substituting the call with the body of book-template is acceptable.
- For listof-book-template, many students forgot to add the recursive call around (rest list-of-books).
- In part (a), many students did not include a contract for their template functions.
- Some students did not understand what a template function was. Please review the lecture slides starting from M06 and see the sample solutions for this assignment.
- In part (d), some students used (listof (cons Str (listof Str))) instead of AuthorIndex in their contract. Although the contract is still correct and no marks were deducted for this, it is preferred to use the name of the data type in contracts instead of its actual type appearing in its data definition.

Question 4

- Some students submitted code that did not run due to syntax errors such as missing parentheses, resulting in a loss of a lot of marks.
- Some students submitted files that were not plain-text, so the correctness grading scripts could not run on them. Please remember to check your basic test results on MarkUs to ensure your file is readable.