# Post-Mortem

Assignment 05

March 3, 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.

## General

- Students are highly encouraged to make sure their code runs before submitting it.
- **Many students were missing examples for base case(s) throughout the assignment. When writing a recursive function, there should be an example for every base case, and an example for at least one of the recursive cases, as discussed in the style guide.**
- Types in contracts should be correctly formatted as per the style guide.
- Many students did not use pure structural recursion. No marks were deducted, but pure structural recursion is encouraged for questions in this assignment. However, for the bonus question, only pure structural approaches were permitted.
- Requirements should apply to the function parameters, not the type of the parameter. For example, if a function parameter named n, which is a `Nat`, must be non-zero, the requirements must state that `n cannot be zero` instead of `Nat cannot be zero`.
- Contracts that consumed or produced lists sometimes had formatting errors. For example, if a function consumes a list of numbers, it should be indicated as `(listof Num)` in the contract **exactly**, and not `listof(Num)`, `list of Num`, `list-num`, `(listof positive)`, `list`, or any other variation.

## Question 2

- Part (b) could have been done without recursion. While many students used recursion, they failed to do it pure structurally. Although no marks were deducted, a pure structural approach is encouraged.
- A few students used `(list Nat Nat Nat)` instead of the given data definition, `Decryptor`, in their contracts. While no marks were deducted, we prefer `Decryptor`. However, using only `(listof Nat)` would be incorrect.
- Instead of defining a helper function that performs recursion on a list of characters and having the main function as a wrapper that used `list->string` and `string->list`, many students did not use any helpers at all, and kept converting strings to lists and vice versa on every recursive call, which indicates bad coding style.

## Question 3

- In part (a), many students forgot to handle the case where no new tags were to be added in `retweet`.
- In parts (b), (c) and (d), many students indicated in their requirements that the values in the produced list must be in the same relative order as they were in the consumed list. Requirements listed in the design recipe should only apply to the arguments given to the function, not to the result of the function.

## Question 4

- Many students had trouble with contracts. Some students chose not not to use the provided data type in their contract, e.g. using `(listof Int)` instead of `(listof Rating)`. In such cases, we prefer `(listof Rating)` but `(listof Int)` is also acceptable as long as the requirement `"all values of ratings-list must be 1 or -1"` is included.

## Ongoing Errors

The following is a list of common errors from previous assignments that were still repeated for assignment 5.

- **Many students are still missing parameter references in their purpose statements. Purpose statements should meaningfully use each parameter name, and the parameter names should be written exactly as they appear in the function header.**
- 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.
- All design recipe components, except for tests, are required for helper functions. Some students did not include these design recipe components for their helper functions.
- Whenever possible, specific equality predicates should be used (e.g. use `string=?, symbol=?, =` over `equal?`).
- Starting a new `cond` expression immediately in an `else` clause is unneeded. Instead, directly check for the next condition in the original `cond` expression.
- Helper functions should not interrupt the design recipe for the main function. Please consult the style guide for more details.