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 6.

**General**

- Many students wrote predicates that followed the form \((\text{cond} \ [X \text{ true}] \ [\text{else} \text{ false}])\) throughout the assignment, where \(X\) is some question. This is unnecessarily complex, as the function body can be simplified to include only the question itself.

- Many students did not properly format their contracts when a function produces more than one type of value. For example, if a function produces either a \textit{Num} or a \textit{Bool}, the contract for that function after the arrow \((\rightarrow)\) should be written \textit{exactly} as \((\text{anyof} \ (\text{Num} \ \text{Bool}))\), or \((\text{anyof} \ (\text{Bool} \ \text{Num}))\).

**Question 1**

- In part (b), many students had an incorrect contract, indicating that the function \textit{get-elem} would produce a \textit{Num}, when it can produce either a \textit{Num} or the value \textit{false}.

- In part (c), some students defined their own version of \textit{length}, when it was clear on the assignment that this was not allowed.

**Question 2**

- In part (a), some students overcomplicated their solution by removing elements from the consumed list in a non-structurally recursive way, or by using sorting algorithms that are not discussed in this course, such as bubble sort or selection sort.

- In part (c), many students had a solution where they would remove all occurrences of an element from the consumed list after the key/value pair was produced for that element. However, that solution would be using generative recursion, as removing an arbitrary number of elements from a list at once is not getting one step closer to the base case, according to the data definition of a list.

- In part (d), almost all students did not include the requirement that the keys (the first of each sub-list) in both the consumed lists are unique.

**Question 3**

- Many students had conditions that directly checked the left and right subtrees in their functions. This was unnecessarily complex, as the recursive calls on the left and right of the consumed tree would check for these conditions as well. Throughout this question, the only place where a check on the left and right was necessary would be in the recursive calls.
• In part (a), some students indicated in their contracts that `height` produces a `Num`, when the function always produces a `Nat`.

• In part (b), some students checked if the list of symbols was empty before checking if the tree was empty. However, if they were both empty, the function would produce an error when calling `node-key` on an empty tree.

• In part (b), many students did not specify in their contracts or their requirements the symbols in the consumed list could only be `'L` or `'R`.

Ongoing Errors

The following is a list of common errors from previous assignments that were still repeated for assignment 6.

• Purpose statements should meaningfully reference each parameter name without repeating information in the contract, and reference each parameter exactly as it is written in the function header. Many students are still writing purposes that do not satisfy both of these conditions.

• Many students were still 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.