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 7.

**General**

- Many students who included multiple helper functions inside a `local` were missing separators between their function blocks. Not only are separators necessary for functions defined at the top level, but they are also necessary for locally defined functions.
- Some students were missing design recipe components for locally defined helper functions. Remember that purposes and contracts are still required for any functions defined inside a `local`.
- Some students were missing brackets in their contracts when indicating that a function either consumes or produces another function.

**Question 2**

**Part b**

- Many students did not comment out their templates, which resulted in mark deductions for black highlighting. It was stated in the assignment that templates should have been commented out.
- Many students had an unnecessary `cond` to check whether the consumed `Trie` would be empty in their `trie-template`. However, this condition will always be false, as a `Trie` structure itself can never be empty. The only template function that should have a check for `empty?` would be the `list-tnode-template`.
- Many students were missing ellipses in their template functions where necessary. In particular, many students did not include any ellipses in their `trie-template`. Template functions should be kept as general as possible, and this includes putting ellipses in the relevant places, so that they can be filled in later.

**Part d**

- Some students kept track of the original `Trie` or list of `TNodes`, and re-recursed through the entire `Trie` once they found a complete word. This approach is unnecessarily complex, as it was only necessary to traverse through the entire `Trie` once.

**Part e**

- Some students had a long solution that did not use mutual recursion. This often lead to an implementation that recursed on the children of the `TNode` in a list of `TNodes`. Using the templates as a starting point would have lead to a cleaner solution that used mutual recursion.
• Almost all students did not include the requirement that the consumed string cannot be empty. The assignment stated that the empty string is not a legitimate word, and thus cannot be inserted into a Trie.

Part g

• Some students attempted to list out all the words in the consumed Trie using list-words, convert each word into a list of characters, and then filter out the words that did not begin with the consumed prefix. However, this solution would be extremely inefficient with a large Trie. Instead, traversing through the tree until the prefix is empty, and then listing all the words at that point, would be a much more efficient solution.

Question 3

Part b

• Many students did not correctly handle the case where the list of functions was non-empty, but the table was empty. Many students simply produced empty in this case, when the function should have produced a list containing as many empty lists as the number of functions in the consumed list.

• Some students attempted to condense their contract by indicating that the consumed list of functions is of type (listof (Num → (anyof Num Int Nat))). However, this implies that one of the functions in the list may have the contract (Num → (anyof Num Int Nat)), when the assignment indicated that each function in the list must have a contract of either (Num → Num), (Num → Int), or (Num → Nat).

Part c

• Some students did not use accumulative recursion when finding the minimum element in the consumed Table, which lead to an exponential blowup.