Question 1

—— For 1 (c), we get one constructor, one type-predicate, and one selector for each field. Therefore, the number of functions is $1 + 1 + 2 = 4$.

—— For 1 (g), appending `empty` to a list does nothing, but appending `(list empty)` adds the value `empty` to the list, which increases its length by 1.

Question 2

—— Structures are not lists! You cannot use `first`, `second`, etc. You must use selectors such as `substance-name` to access fields and the constructor `make-substance` to create a `Substance`.

(a) (b)

—— Use `define` and `(make-substance ...)` to make the constant.

—— Watch out for unnecessary parentheses.

—— 2 (b): some students did not check bounds correctly (i.e. used $<=$ when $<$ was needed, etc.) or produced the incorrect symbol (i.e. `'solid` instead of `'gas`).

—— Some students wrote `substance-mp` or `substance-bp` but did not apply these functions on anything.

(c)

—— There are 5 tests: one for `( < temp mp )`, one for `( < mp temp bp )`, one for `( < bp temp )`, and two boundary cases: `( = temp mp )` and `( = temp bp )`.

(d)

—— Some students missed the `else` case.

—— Use `symbol=?` to compare `States`.

—— Use `substance->state` from 2 (b).

—— Some students compared names, not states.

(e)

—— Again, use `substance->state` from 2 (b).

—— Use `make-matterstate` to construct a `MatterState`.

—— Use user-defined types in the contract ((`listof Substance`) and (`listof MatterState`)).

—— Do not pluralize data types (i.e. it’s (`listof Substance`), not (`listof Substances`)).

Question 3

—— 3 (a): Not simplifying `(not false)` to `true` before applying `fxy · x`.
Question 4

(a)
— Incorrect setup of application of helper (either did not apply the helper on \( n \) when counting down or did not apply the helper on \( 1 \) when counting up).
— Did not produce the correct list in the base case.
— Did not produce the list in correct order.

(b)
— Incorrect comparison for the sum of divisors (many students compared to \( n \) instead of \( 2n \)).
— Missing or incorrect Requires statement in the contract, including failure to reference the parameter name explicitly.

Question 5

(a)
— Many students did not extract each of field of a tv.
— Many students did not extract the movie title using \((\text{first } ml)\).

(b)
— Forgetting to check if \((\text{first } ml)\) is a tv or a string before comparing its value with \( s \).
— Using and to check if \((\text{first } ml)\) is a Movie with the title equal to \( s \), then assuming that it must be a tv otherwise, when it can be a Movie that is not equal to \( s \) (and vice versa with checking tv first).
— Having an extra, unnecessary else case after checking that the first element is neither a string nor a tv (it cannot be anything else).

Question 6

(a)
— Many students incorrectly accessed the name in the Phonebook, resulting in an error for the string< function.
— For the second line, many students wrote \((\text{string<} (\text{first } (\text{first spb})) (\text{first item}))\), resulting in the list being sorted in descending order instead of ascending order.
— For the third line, many students incorrectly cons-ed item onto a recursive call, over-complicating the answer and resulting in unnecessary processing for sorting the Phonebook.

(b)
— Many students incorrectly kept the unchanged phone numbers when the question only wanted a list of the new phone numbers.
— Some students had an incorrect conditional to check if the phone numbers were equal (they either used a more complicated function instead of \( = \) or they compared the names instead of the phone numbers).
— Too many students didn’t connect this problem to the lockstep problem studied in class. That led to issues with the base case doing more tests than necessary, checking for inequalities, and with stepping through the lists incorrectly.
(c)  
— Not making use of the accumulator in the base case.
— Not using the order of the lists to process both at once.
— Not using accumulative recursion.
— Not starting the accumulator at 0.
— Producing a (listof Str) instead of a Nat.

(d)  
— Not including a contract.
— Using (rest (first phonebook)) instead of (second (first phonebook)) to compare the first phone number in the Phonebook.
— Not using (first (first phonebook)) to access the first name in the Phonebook.