Post-Mortem

Midterm 2

October 8, 2019

We normally publish the post-mortem for a midterm after it has been marked and released. Here is a list of common errors provided by the graders for midterm 2.

Question 1

- In part (a), many students stated that the statement produces 2 or 5. Since \( \text{cons} \ \text{elem} \ \text{lst} \) simply adds \( \text{elem} \) to \( \text{lst} \) without modifying \( \text{elem} \) in any way, the correct answer is \( 1 + (\text{length} \ q2) \), which is 3. \( (\text{length} \ (\text{list} \ q1 \ q2)) \) would produce 2 and \( (\text{length} \ (\text{append} \ q1 \ q2)) \) would produce 5.

- In part (b), many students missed the fact that \( (\text{list} \ q1) \) is a list with only one element in it, so \( (\text{rest} \ (\text{list} \ q1)) \) produces empty.

- In part (b), some students added empty as the first element in the new list. The key here is recognizing that \( (\text{append} \ \text{list1} \ \text{list2}) \) adds all the elements of \( \text{list1} \) to \( \text{list2} \), and empty has 0 elements.

- In part (c), some students answered with \( (\text{cons} \ (\text{cons} \ \text{empty} \ \text{empty}) \ \text{empty}) \), which represents \( (\text{list} \ (\text{list} \ \text{empty})) \).

- In part (d), many students forgot a few boxes, especially the boxes corresponding to \( (\text{cons} \ (\text{list} \ 4) \ \text{empty}) \).

- In part (e), many students had only one occurrence of first in their answer instead of two.

Question 2

Part a

- Many students did not check to see if the first element in the list is also a list of length 2 before calling \( (\text{number}? \ (\text{first} \ (\text{first} \ \text{lst}))) \) and \( (\text{string}? \ (\text{second} \ (\text{first} \ \text{lst}))) \).

- Some students used \text{length} which was not allowed for this question.

Part b

- Many students assumed that the list was sorted and thus only checked to see if adjacent pairs in the AL have duplicate keys instead of checking the entire list.

- Some students had trouble with correctly selecting the key from the AL using \( (\text{first} \ (\text{first} \ \text{lst})) \).
Part c

- Many students made errors in their contracts, such as using Any instead of (listof Any).
- Many students used cond in this question instead of using and, not and or. Although a solution with cond is still correct, it is much better to take advantage of the fact that predicate functions return boolean results that can be manipulated with boolean algebra.
- Overall, this question was well done!

Question 3

- Many students forgot to include the requirements that slon and dels are sorted in non-decreasing order in their contract.
- Some students forgot to handle the two base cases where dels and slon are empty (in that order).
- Some students incorrectly recursed on (rest dels) when (first slon) and (first dels) are equal.
- Some students mixed up the recursive calls for when (< (first slon) (first dels)) and (> (first slon) (first dels)).

Question 4

- Many students made various errors in their contract, such as forgetting to include the requirements that the symbols are (anyof 'A 'B), or that the result is a (list Nat Nat). Even though (list Num Num) and (list Int Int) are technically correct, please use the most specific types possible in your contract.
- Some students made multiple passes on the list, which was disallowed by the question. Note that using functions such as length counts as making multiple passes on the list, since they make (at least) one pass on the list as well.

Question 5

Part a

- Some students did not have the correct format expected for a data definition. Please see the sample solutions and the lecture slides for Module 06 beginning from slide 19.
- Many students forgot to include the requirements on weekday and sky in their data definition.
- Some students used Nat <= 100 instead of pop <= 100 in their requirements. This is ambiguous - restrictions imposed on a parameter must use the name of the parameter.
- Some students wrote (make-daycast weekday sky high low pop) in their data definition and specified their types in the requirements instead of writing (make-daycast Sym Sym Int Int Nat).
Part b

- Many students did not meaningfully refer to all their parameter names in their purpose statements.

- Many students named their parameters Daycast or daycast, which is not a good choice since it’s quite easy to confuse it with existing structure and data definitions.

- Many students did not define appropriate constants and use them in their code. For this question, at least 3 constants representing the low temperature, high temperature, and probability of precipitation were expected.

- Many students combined multiple tests into one check-expect. This is discouraged since if the test fails, it is hard to determine what the cause of failure is since more than one thing is being tested at a time. Always try to make sure that one test only tests one specific case where your code could fail.

- Some students did not cover all the required test cases.

- Some students mixed up the order of the design recipe components.

- Some students had trouble appropriately using the selector functions for a Daycast, typically by using something like daycast-sky in their code instead of applying the selector function to the consumed daycast, such as (daycast-sky daycast).

Part c

- Many students did not check to see if the length of the consumed Forecast is at least 2 before using functions like second.

- Many students did not recurse on the entire Forecast and stopped at the first weekend. This is not completely correct since the consumed Forecast may have multiple weekends in it and the function should produce true if any of the weekends in the Forecast are nice.