In addition to the ones provided for assignments, we also publish post-mortems for each midterm after it has been graded. Here is a list of common errors provided by the graders for midterm 2.

General

- It is important to realize that \textit{cons}, \textit{list}, and \textit{append} do different things with lists. Many students used these functions incorrectly throughout the midterm.
- Some students did not take advantage of ordering their conditions to reduce the amount of checking they had to do for later conditions.
- Some students wrote question/answer pairs that would never be evaluated.
- Some students started a new cond expression in the associated answer of an else condition.

Question 1

- Part (a) was poorly done overall. Many students thought that the length of \texttt{(cons q1 q1)} would be 2 or 6, but if \texttt{lst} is a list, then \texttt{(length (cons X lst))} will always evaluate to \texttt{( + 1 (length lst))}, regardless of what \texttt{X} is.
- In part (b), some students did not take the first of \texttt{(rest q1)} enough times.
- In part (c), many students were missing \texttt{“(define q1 ...”} at the beginning of their answer, when the question asked to rewrite the \texttt{definition} of \texttt{q1}.
- In part (d), many students were missing some instances of \texttt{“cons”} and \texttt{“empty”} throughout their answer.
- Part (e) was also poorly done overall. Many students drew a nested box diagram of \texttt{q1} instead of a box and pointer diagram. Many students who drew a box and pointer diagram did not accurately show how the lists were nested in their diagram.

Question 2

- Some students did not match their type of recursion to reflect how the parameters were being changed in the recursive call.
Question 3

- Part (a) was generally well done.

Part b

- Many students forgot to include the contract in their template function. Contracts are always assumed to be required for template functions.
- Some students thought that Quuxes were lists, and included tests for empty?, as well as using first and rest throughout their template. Although many of the templates in this course involve lists, it doesn’t mean that all templates involve lists.
- One cond case should look like the following:
  \[ (\text{boolean? } (\text{quux-foo } qx)) \ldots (\text{quux-foo } qx) \ldots (\text{my-quux-fn } (\text{quux-bar } qx)) \ldots ] \]
  Many students omitted the second \((\text{quux-foo } qx)\).
- The else case in the cond isn’t equivalent to \((\text{quux? } (\text{quux-foo } qx))\), which some students did not realize. That doesn’t handle a case like \((\text{make-quux } 5 \ 5)\), which is still a valid Quux according to its data definition.
- Many students either omitted recursive calls, used structure selectors improperly, or used the wrong names for type predicates.

Part c

- Most students did well on the examples and contract.
- Generally, students had an easier time writing the function than the template, though errors in templates could become errors in the function as well, such as treating Quuxes as lists.
- Some solutions included a base case for Booleans, and recursive cases that relied on it. That technically works correctly for valid Quuxes, but the function violates its own contract by permitting a Boolean (which is not a legal Quux) as an argument.
- Some students did not “trust” recursion, and tried to resolve too many layers of the Quux structure in a single function, instead of obtaining the rest of the answer from a recursive call.

Question 4

Part a

- Many students indicated that the function would produce a “(listof (anyof (list Str Str) Str))” instead of a “(listof (anyof (list Str Str) (list Str)))” in their contract. However, indicating that the function produces a “(listof (listof Str))” was also acceptable.
- Some students forgot to include “anyof” in their contract, or used invalid syntax throughout their contract.

Part b

- In subquestion 1, many students wrote that the function produces empty. Although this is technically correct, the question indicated that quoted list notation must be used in the answer.
- In subquestion 2, many students indicated that the function produces ‘(“Dave”), when it should produce a list with ‘(“Dave”) as its first element instead, i.e., ‘((“Dave”)).
• In subquestion 3, many students thought that Dave should come before Mace, when Mace should come first instead.

Part c
• It was common for students to have the following code in their base cases: 
  \[ \text{cond \ [(empty? lst1) lst2] \[(empty? lst2) lst1].} \] This would produce the correct answer (empty) if both lists were empty, but it does not produce the correct answer when just one of the two lists is empty.

• Many students consed the string “Mace” or (first lst) onto the result of recursing, when “Mace” should be put in a list instead - i.e. (list “Mace”) or (list (first lst1)).

• Some students used quoted list notation incorrectly. While ’(“Mace”) is fine, ’((first lst1)) is not. That will produce a list containing a list, which contains the symbol ’first and the symbol ’lst1. (Try it yourself in DrRacket).

• Some students used symbol=? to compare strings, when string=? should have been used instead.

Question 5

Part a
• In step 1, some students replaced (f x) with f’s body before substituting 5 for x.
• Some students substituted all x’s in the original expression at once, instead of just the first one.
• Some students substituted (f 5) with the result (30), rather than the body of f.

Part b
• Some students missed the fact that taking the first of an empty list would produce an error.

Part c
• Many students missed the fact that q is defined to be (make-s false 5 ’fish) rather than (make-s false x ’fish). Since the question stated that the definitions provided have been completely simplified, the x is replaced by 5 when (define q ...) happens.

• Some students skipped the step of substituting the value of q into the expression. They went straight from (s-a q) to false.

Part d
• The most common error was in step 2, where most students took (rest (cons empty (cons q empty))) as (cons q empty), when q should have been replaced instead.

• Some students replaced q with (make-s false x fish) instead of (make-s false 5 fish). If this was the second time you made this mistake, we let it go.

Part e
• Some students got lost in the trace and produced the list in the wrong order.
Question 6

- Many students believed the order of the base cases could be swapped. However, they cannot, since empty is a subset of empty.

- Some students used length in their solution, which made the solution, at best, inefficient and, at worst, incorrect.

- Many students recursed if \((< (\text{first set1}) (\text{first set2}))\) was true. However, since set2 is increasing, then \((\text{first set1})\) will never be found in set2, and thus, set1 is not a subset of set2. Recursing will potentially provide an incorrect answer in this case.

Question 7

Part a

- Some students missed parentheses when creating their constant. Some students did not define a constant at all. Some students made their constant with strings and not symbols.

- Some students misused quotes around symbols and lists.

Part b

- In the base case of an empty stock market, the function should produce the ticker and price as a tuple wrapped in a second list. Many students forgot to wrap their answer.

- In the base case, many students produced empty and not the tuple of new ticker symbol and price.

- Many students misused cons. \((\text{cons ticker-symbol stock-price})\) is incorrect. It should either be \((\text{list ticker-symbol stock-price})\) or \((\text{cons ticker-symbol (cons stock-price empty)})\).

- The case where you find the stock symbol should (usually) not recurse on the smaller list. If you do, then you probably will insert the stock symbol twice.

- Some students lost easy marks because they named parameters poorly. In this course, we stick to names with lower-case letters separated by dashes.

- Some people attempted to use append to insert \((\text{list ticker-symbol stock-price})\) into the stock market. This is allowed, but \((\text{list ticker-symbol stock-price})\) again needs to be wrapped in another list, or the resulting list will not be an association list.

- In the recursive case (when the stock is not found), many students did not cons the first element of the stock market with the result of the recursive call. Unless you cons the first element you will lose your stock market and the economy will crash.