Midterm Post-Mortem

Q1 – Students can run the code themselves to verify the output or error, and are encouraged to ask assistance if they don't understand the answer.

Q2 – Common errors included:

- not correctly substituting all arguments for parameters in a function call
- incorrectly substituting all constants at once
- incorrectly evaluating right to left instead of left to right
- combining or skipping steps

Q3 -

a) overall – some students didn't seem to understand that they were supposed to write one thing in each blank
   - my-and: The most difficult part was in identifying the first clause (cond [a b]), that is, if a is true, the value of (my-and true b) simplifies to the value of b.
   - my-or: A fair number of students wrote (cond [a true] [b true]). This does indeed return the correct answer if (or a b) is true. However, if both a and b are false, there is an error.
   - not: Most students wrote out the truth tables, but had difficulty making the jump to writing the code.

b) Many students didn't seem to understand this question. When my-and and the built-in special form and are both called with simple Boolean values (true and false), they will give the same answer. However, if they are passed more complicated values, they may behave differently. For example, suppose we call (my-and false (= 1 (/ 2 0))). The second argument will be simplified to a divide-by-0 error. However, due to short-circuit evaluation, calling (and false (= 1 (/ 2 0))) will evaluate to false, and the divide-by-0 error never happens because the second argument is not simplified.

Q4 -

a) Part A caused the most problems. The question asked for a data definition and many students provided a function. Those who defined a function that included empty and an appropriate recursive case got partial marks. Another common approach was to use first and rest instead of (cons Whatzit (listof Whatzit)).

b) Part B was mostly well-done with only the occasional student making one or more careless syntax errors.

c) In part C, some forgot to add 1 for non integers.

Q5 -

a) The biggest issues were simply related to not remembering what parts were necessary in a structure definition or a data definition. Most students included a complete or partially complete structure definition, but some left out the data definition altogether, or forgot how to specify the types for the fields. Note also that the struct name should be lowercase, and the type name in the data definition should be uppercase.
b) Many students had trouble remembering how to format a template. Some left out the contract, or got the types in the contract wrong. Another common problem was to omit the Route argument from the template function, or to require separate Nat, Str and Bool arguments. The function should consume a single argument, which is then passed in to the three selector functions.

c) Generally this was reasonably well done. The two biggest issues were either not actually defining a constant (i.e., leaving out the "define" part of the solution), or leaving out the constructor (e.g., "make-route").

d) Most students struggled with this question. Some didn’t know what a type predicate was. Others did not come up with the correct name for the function, or didn’t know the correct contract (every type predicate consumes Any and produces Bool). Some students thought that because the question refers to a predicate, it was somehow tied to the overnight? field of the route structure; the two are entirely unrelated.

Q6 –

a) Many students did not use the type (listof (anyof Nat Str)) in the contract. The syntax on this question was strongly enforced.

b) The function implementation was generally well done. The most common errors were failing to check that (first lst) was an integer before calling the even? or remainder function, or, similarly, failing to check that (first lst) was a string before calling string-length.

c) Examples were very well done. The most common error was failing to include an example for the empty list, which would be expected for examples following our style guide (module 05, slide 33).

Q7 –

a) Common errors included:
   - Placing brackets around the constant name
   - Using make-midterm or midterm-date as a constructor
   - A few incorrect dates

b) Common errors included:
   - Placing brackets around the constant name
   - Incorrect selectors (midterm-day etc.)
   - Using constructors instead of selectors (make-midterm, make-date etc.)
   - Not providing arguments to selector functions

C) Common errors included:
   - Not providing arguments to selector functions
   - Treating the Boolean variable as a function (placing brackets around variables, in general)
   - Forgetting the "/" in-between strings
   - Using constructors instead of selectors
   - Using number->string incorrectly (i.e. only using once for three numbers, when needed for each number)
   - Using string-append incorrectly (i.e. using multiple times for five strings, when needed only once)

Q8 –
a) Many students did not follow the format for writing the contract (e.g. types were not capitalized, no ;; to indicate the contract is a comments block, missing “requires:”, parameters were not named in the requirements section, etc.)

Some students used Int as a type for day (instead of the better choice Nat) then further specifying that 1 <= day <= 7.

Some students combined the purpose with the contract or added unnecessary requirements to the contract (e.g. specifying that vip? is one of true or false).

b) Students did well on this question. Some students used check-within (check-expect would have been sufficient for almost all provided tests). (No marks were deducted for that.)

c) Students did quite well with the function implementation. The most common mistake was checking the value of vip? (e.g. (vip? true), (vip?), (true? vip?), (= vip? true), etc.). The second most common mistake was multiplying by discount rather than 1-discount. Some students did not use constants.