

POST MORTEM
CS 135 Fall 2025 Midterm

Question 1: Short Answer

Mistakes:

- c) was frequently incorrect; a reminder that big O is an *upper bound*, and $n^3 \ll 2^n$ asymptotically.
- Similarly, e) was frequently answered **polynomial** instead of **exponential**. In `blah`, when following the path to the **else** case, you encounter 2 recursive calls (`blah (rest lst)`), and because one is in the question, it will always be called (even if the second **cond** is not the one chosen)
- Less frequently, but still notably, many students forgot the format of `check-within` for d). It's (`check-within function target tolerance`)

Question 2: Racket Expressions

Mistakes:

- a) was mostly well done, though once in a while students got confused by the middle `cons` and gave a different number than 5.
- Students commonly forgot that lists can contain all different types, *including booleans*, and thus incorrectly said that c) was an error.
- Many students missed the fact that the `cons` provided in b) had 3 inputs, when `cons` only allows for 2.

Question 3: Stepping Problems

Mistakes:

- Students commonly mixed up `x` and `y` in a), causing them to get the wrong answer.
- Students commonly misidentified the first step in a); you first need to resolve constants in the function call before expanding.
- Similarly, in all 3 parts, students frequently applied constants before they should.
- In c), students got rid of the **cond** substantially faster than they should have; step 2 should still contain **cond**.

Question 4: Short Functions

Mistakes in Part a):

- Many students only checked whether the argument was a list and whether the rest of the rest of the list was empty. This fails to account for the possibility of the argument being the empty list or a list of length 1, since calling `rest` on `empty` produces an error.
- Some students seemed to be trying to use the structure of **cond**, though **cond** was disallowed for this question. The question is possible with a single boolean expression.

Mistakes in Part b):

- Many students defined the input and/or output types as `Any`. This neglects the fact that if the input is a number, it must be a natural number, since we perform recursion on it in `mystery-machine`.
- Many students wrote `List`, or something of similar spirit, as one of the output types instead of (`listof Nat`).
- Many students wrote `Int` instead of `Nat` for one of the input types, as well as the (`listof Nat`) in the output types. Inputting a negative integer into `mystery` breaks the recursion in `mystery-machine`.

- Many students wrote `Int` as one of their output types. This is technically correct as `mystery` outputs `-5` when the input is a `Sym`, but one can notice that the *only* number that `mystery` will output is `-5`. Thus, for full marks, we can specify the exact output `-5` instead of the broader `Int`.
- A purpose statement or requires clause was unnecessary for this question as we only asked for a contract (No deduction). Many people wrote `Int` as one of their consumed types and wrote a requires clause requiring that the `Int` was ≥ 0 . This can be concisely represented in the contract itself as `Nat`.

Mistakes in Part c):

- Many students neglected to test the “trivial” case when the input is 0.
- Some students wrote incorrect test cases such as `(check-expect (mystery 3) (list 3 2 1 0))` or `(check-expect (mystery 3) (list 3 2 1 empty))`.
- Some students wrote incorrect test cases such as `(check-expect (mystery -5) error)` or something similar. We don’t have the ability to test for errors like this; this would just crash in DrRacket. (The function `check-error` can check for errors, but it is **not** on the Allowed Constructs.)

Question 5: `apply-lists`

Mistakes:

- Many students tried applying the symbols directly as functions instead of using a helper function to determine which function to apply.
- Many students recursively added a `0` to the list of numbers when it became `empty` which went against our current rules of recursion.
- Many students tried solving the problem through an accumulator, which effectively reversed the answer, but failed to reverse it back.
- Many students failed to realize that when the list of `Op` became `empty`, the list of numbers should be produced as the result.
- Many students did not consider the case where the list of numbers became `empty` with more than one `Op` left.

Question 6: `list<?`

Mistakes:

- When `lst1` and `lst2` had the same length, many students did not short circuit to produce `false` when `(> (first lst1) (first lst2))`.
- Many students did not compare the lengths of `lst1` and `lst2` before performing recursion on the lists.
- Many students failed to produce `false` when both `lst1` and `lst2` are `empty`.
- Some students did not perform recursion when `lst1` and `lst2` were the same length.

Question 7: `del-list`

Mistakes:

- Many students who used accumulative recursion forgot that the produced list is in reverse order. As a result, these students forgot to reverse the list at the end to preserve the sorted order.
- Some students also didn’t name their parameters as per the question specification (e.g. used `lst1` and `lst2` or some other variation instead of `sorted` and `dels`).
- Some students also used `symbol=?` instead of `=` for comparing numbers.
- Some students forgot to account for both base cases of when `del` or `sorted` was empty.

- Some students didn't properly recurse towards a base case in their recursive calls.

Question 8: Be a Star!

Mistakes in Part (a):

- Many students used `listof` instead of `list` in the contract for `Star`.
- Some students did not use `Requires` clause to bound the 2nd and 3rd field of a `Star`.
- Many students did not use the functions `second` and `third` to simplify implementation for the accessor functions. (No deduction).
- Some students did not understand what a constructor and accessor function does, and wrote predicate functions instead.
- Some students define the colour of a star to be a `Sym`, which makes it non specific to the only 3 colours.

Mistakes in Part (b):

- Many students used `(listof Any)` instead of `(list Any Any Any)` in the contract.
- Many students used incorrect data types in the contract.
- Many students failed to check whether the first element was a symbol before applying `symbol=?`.
- Many students failed to check whether the second element was an integer before checking whether it was ≥ 3 .
- Many students failed to check whether the third element was a number.
- Some students misunderstood or oversimplified the question. As a result, these students wrote a function that takes in a `Star` and check if it's a `Star`, or takes in 3 arguments that are `Star`-like instead of a list.

Mistakes in Part (c):

- Many students did not handle, or incorrectly handled the `'blackhole` case.
- Many students have repetitive code blocks that check for different colours, instead of having a helper function. (No deduction).
- Some students did not use the accessor functions they wrote in Part (a). (No deduction).