CS145 F25 Midterm Postmortem

November 3, 2025

The mean and median for the midterm was 78.6% and 83.6% respectively. The following is a list of common mistakes made on each question.

Q1:

- Solving for n_0 such that $f(n_0) = cg(n_0)$ case and then immediately claiming that $f \in \mathcal{O}(g)$ with (c, n_0).
- In proving $f \notin \mathcal{O}(g)$, choosing specific c or specific n_0, instead of proving that you get a contradiction no matter which c, n_0 you choose.
- Not solving from definitions.
- Proving that $f(x) \leq cg(x)$ implies some lower bound on x, when you actually need that the lower bound on x implies $f(x) \leq cg(x)$.

Q2:

- Writing unnecessary helpers.
- Attempting to use the built in `remove`, even though it was not discussed in lecture. Furthermore, most students who used the built in `remove` did not realize that `remove` will delete the first instance of an element from a list, not the element at a given index.
- Attempting to use `take` and `drop` without definition, even though they aren't built in to intermediate student with lambda.
- Not reversing lists when appropriate.
- Using `list-ref` or `append` in ways that led to inefficient solutions.

Q3:

- Reading the provided list in the wrong order, that is, making the first element of the list the least significant digit instead of the most significant digit.
- Not checking if the current digit is true or false.

Q4:

- Implementing `nextkey` and `prevkey` as if they were working on a normal list rather than a circular list, that is, they did not circle back around the list when the input was the first or last element in the list.
- Attempting to use `list-ref` to get in the index of a key.
- Using `cons` where `append` was appropriate, or vice versa.
- Forgetting to reverse an accumulator where appropriate.
- Inefficient solutions, often caused by repeated calls to `list-ref`.

Q5:

- Having `ea` and `sm` in the wrong order in part a.
- Using incorrect syntax to access struct fields.
- Using `append` in part c, giving an $\mathcal{O}(n^2)$ solution.

Q6:

- Failing to use tail recursion.
- Missing the base cases for $n = 0$ or $n = 1$.
- Swapping the order of A and B during recursive function calls.
- Off by one error in the final result, mostly caused by returning the incorrect option from $\{A, B\}$.

Q7:

- In part a, claiming that the given code produces an error if either subtree is empty, since it tries to take `(node-key (node-left t))`. However, the `or` statement before ensures we are never taking the key of an empty node.
- Missing the empty tree case in part b.

Q8:

- Producing the incorrect final value in part c.
- Claiming that part e produced an error.

Q9:

- Not using ALFs.
- Using incorrect syntax for `foldr`.
- Not using `cons` in part a.
- Using generics in the contract of part b.
- In part b, using `filter` correctly, but then just outputting the resulting list, rather than doing any further checking on it.
- Writing the function body instead of the contract for part d.