Guide to Cheating and Academic Integrity: see link here outlining the academic integrity guidelines for this class

Each midterm typically has an end-user license agreement (EULA) that you must agree to. You will save time if you review this ahead of time and ask any questions before exam day. (It’s better to be safe than sorry!) Typically, the EULA asks you to verify that you will not distribute your work or any material/questions within the exam document to anyone.

Quick cheat sheet for academic integrity:
- Asking other students/TAs/Prof Donald general questions about the homework (or concepts covered in the HW) that clear up misunderstandings (✅)
- Asking other students for their code or copying off another student’s paper (🚫)
- Asking clarifying questions about what will be on the exam (✅)
- Attempting to find out (through any means) ahead of time what specific types of questions will be on the exam (🚫)

Office Hours Tip: It is a good idea to review problem sets/SAs ahead of time and come to office hours with a list of detailed questions. It also helps TAs help you more effectively (they don’t have to guess your question for you). Please avoid asking the TAs to solve the problem for you — it delays your ability to learn!

Note: Solutions to P.S. 4, which contains several of the topics covered in midterm 2, will soon be available on the course website!
What do I actually have to know for Midterm 2?

- Implementing functions in Scheme that utilize important operations such as foldr, map, etc.
  - Understand infinite streams and how to write Scheme functions that take in infinite streams as arguments, rather than finite sets
- Set Theory
  - Bijections, injections, etc. Onto vs. one-to-one (1-1) functions
  - Combining/working with sets (understand set notation, unions/differences/cartesian products/intersections…)
  - Countability, diagonalization, etc. Know the important theorems related to these.
- Standard Induction (proof by induction) - see last guide
- Structural Induction (S.I.)
  - Claim
  - Base case (with substitution model, S.M.)
  - Induction Hypothesis (I.H.)**
    - How does this change in comparison to plain induction? Consider the SA problem where we applied structural induction to a set data type and assumed that P holds for subsets s1 and s2…
  - Inductive Step (I.S.) - must utilize I.H. and (often) S.M.!
- Halting Problem
  - Determine if a variation of the Halting Problem is possible
  - We proved in class that safe? cannot exist. In the homework, we used this same logic to show that halts? (a similar function) also cannot exist. Be sure to understand these proofs (e.g. we wrote safify using safe?).
- Implementing an infinite stream by writing a Scheme function
  - Write a specialized infinite stream (a series of specific numbers) by defining a lambda function that utilizes common stream functions such as add-streams, scale-stream, stream->listn, ones, integers, etc.

Any tips for each of these topics?

- Implementing a function in Scheme
  - Think of each operation (filter, map, accumulate, append, etc.) as tools in your tool box that you can use to write more complicated functions. In which situation should you use accumulate? What about filter? Consider this carefully.
  - Memorize and deeply understand the source code for these operations (and make sure you can recall them from memory so you can use them when implementing your own functions)
  - We now know additional operations such as foldr and foldl. Make sure you know what the difference is between these.
- Set Theory
  - Memorize the definition of a bijection, an injection, etc.
- Memorize the definition of an onto function (every \( y \)-value has corresponding \( x \)-value) and a 1-1 function (no two \( y \)-values have the same corresponding \( x \)-value)

- **Structural Induction (S.I.)**
  - Understand the difference between standard induction and structural induction well.
  - Study the SA problem where you had to write a structural induction proof. Understand it inside and out.

- **Implementing infinite streams**
  - Study P.S. 4., this time writing out the answers by hand. Understand why the solutions are correct.
  - Memorize and deeply understand the source code for the important stream operations (and make sure you can recall them from memory so you can use them when implementing your own functions). Some good ones to study: stream-map, add-streams, mul-streams, etc.
  - Know the stream-equivalents of common operations such as cons, null?, car, cdr, etc. (stream-cons, empty-stream?, stream-first, stream-rest…) Make sure to use these instead when you’re implementing functions.

**How should I study for the exam?**

- **DO** reread the lecture notes and handouts (redo any example problems you encounter!)
- **DO** review the SAs and problems from recitation, as well as the solutions released for related P.S. questions.
- **DO** practice writing code by hand so you’ll be prepared come exam-time.
- **DO** come to the Halloween lecture so you’ll have a chance at extra-credit 🎃👻
- **DO** ask for the blessing of the lambda gods the night before the exam.