Each 6.1800 lecture will come with an outline. You can fill this in during lecture, after lecture, or not at all — it's entirely up to you how you use it. The goal of these outlines is to help you understand the main points that you should be taking away from each lecture. In some cases we will also include examples of things you should be able to do after each lecture.

In the past, these outlines have proved to be an effective tool for studying for the exams. Note that the outlines are ***not exhaustive****; there will be topics and nuances in lecture that aren't captured by the outline.*

Lecture 18: Isolation

- Today we're working with isolation. What does isolation mean at a high level? (We'll make it more precise as we go on)
- Why would running schedules serially (not in parallel) have poor performance?
- Conflict serializability
    - What does it mean for two operations to conflict?
    - How do we create a conflict graph for a schedule of transactions?
    - What does the conflict graph tell us about the conflict-serializability of a schedule?
- Two-phase locking (2PL)
    - What is the goal of 2PL?
    - What are its two phases?
    - How does 2PL work? Both the initial version and the version with reader/writer locks.
        - Why does adding reader/writer locks improve performance?
    - 2PL can lead to deadlock. What can we do when that happens?

You should be able to do the following after this lecture:

- *Given a schedule of transactions, determine the final written state of any variables, as well as the results of any intermediate reads.*
- *Given multiple transactions, determine the conflicts among them.*
- *Given a schedule, determine the order of the conflicts*
- *Given a schedule, draw its conflict graph, and determine whether the schedule is conflict-serializable*
- *Given a schedule, apply locks according to 2PL (both the initial version and with reader-/writer- locks)*
- *Given a schedule that uses locks, determine whether it adheres to 2PL (both the initial version and with reader-/writer- locks)*