Agenda

- **Review the `condition_variable_any` and its contribution to the solution to the Dining Philosophers problem we worked through during the final 15 minutes of Friday's lecture.**
    - In my opinion, the condition variable if the most difficult of the multithreading directives to understand.
    - I want to do my share to ensure you understand condition variables now instead of allowing you to wait until they're needed for later assignments.
- **Introduce the `semaphore`**
    - The most recent solution to the dining philosophers problem enlisted the services of a **`mutex`** and a **`condition_variable_any`**. They collectively provide what can be best described as an integer with atomic increment, atomic decrement, and the added restriction that the integer can never go negative. Any attempt to decrement a 0 prompts that thread to block until some other thread increments it.
    - The counting variable specific to dining philosophers represents a limited resource shared among many competing threads—in essence, a limited number of permits allowing philosophers to eat.
    - We can and will generalize the idea of a counting variable by defining a **`semaphore`** class that encapsulates an integer, provides atomic increment and decrement operations via **`signal`** and **`wait`** methods, and indefinitely blocks a thread trying to decrement a **`semaphore`** surrounding a 0.
        - Conceptually, the **`semaphore`** allows us to model a shared resource—a number of remaining permission slips, a number of

remaining file descriptors, a number of remaining network connections, etc—while insulating us from the complexities that come with coding with **`condition_variable_any`**. **`condition_variable_any`**s are more general than the **`semaphore`**s implemented in terms of them, but a large percentage of synchronization needs can be expressed in terms of the **`semaphore`**, which in my opinion is easier to understand.

- Many modern languages provide native support for threading and synchronization.
  - Java, in particular, has supported threading and condition variable-style locking since the beginning. It eventually added a **`Semaphore`** class in the early 2000's.
  - Honestly, I'm not sure why C++11 decided to exclude the **`semaphore`**, but I think it's easier to work with than **`condition_variable_any`**s, so I'm going to introduce it so we can go forward pretending that it's just part of the C++ language.