### Lesson 9: Concurrency in Rust

* **Reading**: “The Rust Programming Language,” Chapter 16, alongside select blog posts and official documentation on advanced concurrency patterns.
* **Assignments**: Construct a concurrent data processing pipeline that uses channels for inter-thread communication and synchronization primitives to manage shared state.
* **Preflight**: Review the basics of thread creation and the Send and Sync traits.
* **Lesson Goals**:
  + Deepen understanding of Rust’s concurrency model, focusing on ownership and type checking at compile time to prevent data races.
  + Learn to use threads, message passing, and shared state in concurrent programming.
  + Implement safe concurrent code using synchronization primitives like mutexes and channels.
* **Motivation**: Mastering concurrency is essential for developing efficient, scalable software that can take full advantage of modern multi-core processors.
* **Lecture**:
  + Exploration of Rust’s threads, including thread spawning and joining.
  + Message passing vs. shared state: comparing concurrency models.
  + In-depth discussion on synchronization primitives: Mutex, RwLock, Arc.
* **Lab**: The data processing pipeline assignment will challenge students to apply these concepts, emphasizing safe, concurrent modifications to shared data and efficient inter-thread communication.