Syllabus

1) Introduction, Logistics, and Why Library Design
   - What "Library Design" means.
   - Usability and Interface
   - Readability
   - Efficiency
   - Models of Libraries: STL, IOStreams, Boost, Qt
2) Exceptions: The Last Major C++ Construct You'll Learn
   - Why, how, what, when to throw and catch exceptions
   - Efficiency considerations
   - Exception Specifications and Why They're Evil
3) Transactional Programming: Exception Safety and Design
   - The Abrahams Exception Guarantees
   - The pImpl idiom
   - When not to throw, and when not to catch
4) Policy Based Class Design
   - STL Allocators
   - Decomposing Classes into Policies
   - Problems with the Allocator model
   - Template Template Parameters and their Failures

Assignment 1 Out: smart_ptr<>

5) Trait Classes and Full Template Specialization
   - char_traits and iterator_traits in the STL
   - Implementing Traits

6) More Traits: Partial Template Specialization
   - Generic Traits
   - Class Selection
   - The Substitution Failure Is Not An Error (SFINAE) idiom

7) Template Metaprogramming I
   - Types, Values, Functions, and Memoization.
   - Recursion, Conditions and their ilk
   - Metaprogramming as a LISP dialect

8) Template Metaprogramming II
   - Currying and Lambdas
   - Sequences and Iterators

9) Optimizations and Pessimizations
   - Type-selection
   - Overuse of Templates
   - Copy-on-Write and Shared Strings: An Optimization?

10) Usability: Interface for the Client
    - Expression Templates
    - Domain Specific Embedded Languages
    - Implementing Compile-Time Regular Expressions

Assignment 2 Out: Small library design project (team assignment)