Computer Science Introductory Course Topics

GEN_ENG 151 (note, all topics in the context of **one** language; represents AP CS A content at a 5 level)

- Basics of programming in **one** language (usually Java)
    - This class will assume you are **already** used to designing, developing, analyzing, and documenting code *in some object-oriented programming language.*
- Variables, Types, Objects
- Boolean expressions
- Arrays, ArrayLists, and Nested Arrays
- Iteration
- Abstraction + Basic Class design and implementation (constructors, instance variables, encapsulation, self/this/etc.; methods)
- Subclasses/superclasses (inheritance; polymorphism)
- Basic documentation and testing
- Basic recursion
- Introductory ethical / social thinking around CS

CS 180

- Sets and set operations
- Inductive definition of sets
- Procedures as data
- Recursion
- Functional programming
- Trees and tree depth-first tree traversal
- Types and type signatures
- Specifications
- Inheritance vs. delegation
- Unit testing, test-driven development
- Debugging and reasoning about program failure
- Modularity
- Managing state

CS 208

- Asymptotic performance analysis (worst case, average case, amortized complexity)
- Basic sequence structures (arrays, linked lists, dynamic arrays)
- ADTs: stacks, queues, dictionaries, etc.
- Sorting algorithms
- Hash tables and hash
- Tree and graph representations
- Tree walks and graph search
- Search trees, including self-balancing trees

- Shortest path algorithms, minimum spanning trees
- Priority queues, binary heaps
- Disjoint sets (union-find), including path compression
- Data design
- Relational model

CS 211
- C and C++ language, including functions, parameter-passing, structs, classes, pointers, templates, standard containers and algorithms, iterators, inheritance, polymorphism
- von Neumann machine model (code, memory, references vs. pointers, stack vs. heap, static and dynamic memory allocation)
- Basics of the Unix tool chain (g++/gcc, make, gdb, valgrind, etc)
- Basics of software development process (git, github, etc)
- Unit testing
- Debugging medium-sized programs (1,000+ lines of code across multiple source files)
- Reasoning about and debugging in languages that aren't type- or memory-safe, i.e. memory leaks, uninitialized pointers, buffer overflow, etc.