CT 100 Learning Objectives

1. General Computation
   - be familiar with the history of the modern digital computer from the 1950’s
   - be able to apply Moore’s law as a measure of computing advances
   - understand the limitations and advantages of digital devices by comparison to analog
   - appreciate the computer as a repository of data
   - understand that data is represented as sequences of bits
   - know the relative sizes of byte, kilobyte, megabyte, terabyte and petabyte
   - be able to translate decimal numbers to binary and binary to decimal
   - understand that data has alternative forms (numeric, textual, code, image, sound)
   - understand that different data forms use different, largely incompatible encodings
   - be familiar with the concept of data compression and the difference between lossy and nonlossy compression
   - understand different forms of computer storage and the relative size and performance differences between memory and secondary storage
   - understand the process of program compilation and execution (source code and executables)

2. Logic
   - understand that one bit of data may also store a Boolean value (true or false)
   - be able to analyze a Boolean expression involving the operations of NOT, AND, OR, IMPLIES and EQUIVALENCE
   - be able to express factual English sentence in the form of propositional logic
   - be familiar with logical expression applications in expressing software requirements
   - be able to express database queries at the level of propositional logic

3. Problem Solving
   - understand the nature of software development as a problem solving activity
   - be able to apply divide and conquer strategies to basic problems
   - be able to develop story boarding for solving real world scenarios
   - understand the problem solving approach of prototyping and be able to apply prototyping to implementing simple sequential algorithms
   - be able to translate simple story boards into algorithms involving objects and a solution as a sequence of method calls

4. Control
   - understand the concept of software and program execution
   - be able to write simple programs involving sequences of methods applied to different objects
   - understand that algorithms involve choices and choices take the form of selections involving logical conditions
   - understand that algorithms often involve repetition
   - understand how algorithms are modularized

5. Models of Computation
   - know the definitions of basic digraph terminology, including nodes, vertices, arcs, cycles, walks, and paths.
   - understand the basics of using graphical abstractions as models of computation
   - know the form and function of flow chart elements for imperative statements, selection and repetition
   - be able to express simple algorithms using flow charts
   - understand the concepts of computational state, events and actions
   - be able to model simple, sequence algorithms of ten or less states
6. Organizing Data
- be able to select meaningful names and understand why this is importance for such things as URLs, files, variables, and so forth.
- understand the role of unique naming and the concept of case sensitivity
- be able to craft path names for files organized into file trees
- understand the concept of direct access and indexed data retrieval
- understand the concept of sequential, linked, data retrieval
- know the difference between linear and non-linear data organizations
- understand the concept of data organized in tree structures and applications in classification and analysis.
- understand the concept of 2-dimensional data layout and tabular information retrieval

7. Code is Data / Data is Code
- understand the von Neumann principle that computers can be used to store both data and code
- be able to craft simple spreadsheet
- understand the concept of 2-dimensional data layout and tabular information retrieval
- understand the relationship of discrete functions and tables
- appreciate the role of patterns in software development and the use of software for pattern matching of strings

8. Performance and Limits of Computation
- recognize that performance can be characterized by a count of operations that are performed
- understand the distinct between best case and worst case analysis and the importance of the latter
- understand the limitations of benchmarking vs. analytic analysis
- be able recognize the difference between tractable (polynomial) and brute force (exponential)
- understand that many security mechanisms rely upon algorithmic performance
- recognize that not all things are actually computable (Halting problem)

9. Verification and Validation (optional)
- understand that software correctness only makes sense with respect to specifications
- be able to analyze a small programming problem, the outcome of which is a set of software requirements
- understand that proving program correctness is sometimes a possibility, but not often a reality.
- understand that software correctness is usually accomplished by way of software tests
- realize the limitations of software testing
- be able to craft test cases using simple boundary condition analysis

10. Concurrency
- understand that software correctness only makes sense with respect to specifications
- understand the distinction between sequential and concurrent execution.
- learn how sorting can be accomplished concurrently with a sorting network.
- understand the problem of data integrity in a concurrent environment and some common solutions to data corruption avoidance.
- understand the potential for deadlock and strategies for avoiding deadlock.

11. Security
- understand the basic tenants of security - integrity, confidentiality, and availability.
- realize that complete mitigation is rarely possible, but that a certain measure of security is possible using mechanisms for deterrence, deflection and detection/recovery.
- know basic security terminology, including vulnerability, asset, security system, mitigation, virus, spoofing, spam, sniffing, DOS and CERT.
- understand the proper use of security mechanisms including anti-virus software, passwords, encryption, file backup and firewalls.
- understand the potential for deadlock and strategies for avoiding deadlock.