CS 2500: Course Charter & Syllabus

Jay Aslam, Will Clinger, David Van Horn
College of Computer and Information Science
Northeastern University

February 21, 2012

1 CS 2500 Course Charter

Course Title: CS 2500 *Fundamentals of Computer Science 1* (4 SH)

Course Description: Introduces the fundamentals of program design via study of increasingly complex data definitions: structures, containment, unions, self-referential, mutually referential. In addition, the course studies design via generative recursion, accumulator-style functions, and mutation of variables and structures. The course also introduces models of computations, which explain what programs do when they are applied to inputs.

Prerequisites: The course requires no previous experience in writing computer programs, and only assumes mathematics at the level of high-school Algebra. Students are expected to have an interest in learning about Computer Science in depth.

Textbooks: *How to Design Programs*, Felleisen, Flatt, Findler, and Krishnamurthi, MIT Press, 2000.

Topics Covered: Computers and the nature of computation. (Formal) models of computation. Functions; function definitions. Conditional computation structures. Structure definitions. Modeling information with data. The structure of data: composite, containment, union, self-referential, mutually referential, cyclically defined. Program design and design recipes: design based on the structure of data. Extensional equality of structured data. (Automated) testing. Accumulators. Designing abstractions. Generative recursion.

Course Outcomes: Upon completion of this course, a student should be able to *design programs*, and, consequently, have some understanding of the fundamentals of computation. In particular, the student should be able to: Complete data analysis and definition for problems that require the use of composition, containment, union, and self-reference. Design programs to solve problems that consume and produce data structures based on composition, containment, union, and self-reference. Design test suites for these programs and evaluate the outcomes. Design simple abstractions and test the design. Reason about the performance characteristics of programs in a basic way.

Measurement of Course Outcomes: Outcomes are measured and verified through: Weekly programming homeworks. Daily short paper-and-pencil quizzes. Two to four major exams. Inclusion of student's work in an electronic portfolio.

2 CS 2500 Sample Syllabus

The course sequences through the following topics:

- overview, intro to simple arithmetic calculation
- atomic & compound data:
  - integers
  - booleans & conditional computation
  - records
- representation: information vs. data
- data-driven programming
- testing
- unions
- self-referential unions 1—recursion
- self-referential unions 2—list processing
- designing larger programs: top-down design; iterative refinement
- recursion in all forms: trees, s-expressions, forests
- abstraction: interfaces and implementations
- abstraction: parametric data definitions and contracts
- notions of equality
- local name scope; simple asymptotic complexity
- abstracting control with higher-order functions
- constructing higher-order functions with lambda
- infinite data structures and computations
- generative recursion: root finding, sorting
- generative recursion: graphs & search; accumulators
- review and wrap up