Computer Science 1500 Syllabus Spring 2019

Computer Science 1500: Program Design
Goals: Program Design will provide a hands-on introduction to procedural and object-oriented programming with an emphasis on program design. By the end of the class, students should demonstrate the ability to (1) identify from a problem the appropriate programming concepts for solving it, (2) read and form a high-level understanding of existing code, (3) design and write new code to solve previously unseen problems, and (4) maintain existing code as faults are discovered or requirements change. In all such tasks, they should also demonstrate appropriate use of documentation, version control, and basic verification techniques.
Class: 6:00 - 7:15 PM, Monday and Wednesday in Olin 132
Lab: 7:30 - 9:00 PM, Wednesday in Olin 132
Materials: Think Python by Allen Downey, which is available in digital form at http://www.greenteapress.com/thinkpython/ or as a paperback from the bookstore.

Instructor: Aaron Hall (chall@nebrwesleyan.edu)
Office hours: 7:00 - 8:00 PM, Monday in Olin 132 or by appointment

Grading: Instead of traditional, points-based grading, this course uses specifications grading, which means that each letter grade has an associated set of competencies, you are given repeated opportunities to demonstrate each competency, and once you have demonstrated all of the competencies associated with a grade level, you have earned that grade for the course. This system is designed to give you more control over your final grade and your workload, make the grading process more transparent, and more accurately reflect your learning.

In particular, final grades will be assigned according to the following rubric:

- Students who earn an A will demonstrate mastery of all of the basic, intermediate, and advanced competencies listed on the next page, show that they can draw on multiple competencies to solve new problems without guidance, and give consistently clear explanations of their thought process.
- Students who earn a B will demonstrate mastery of all of the basic and intermediate competencies, show that they can draw on multiple competencies to solve new problems with minimal guidance, and give fairly clear explanations of their thought process.
- Students who earn a C will demonstrate mastery of all of the basic competencies and show that they can draw on multiple competencies to solve new problems with some guidance.
- Students who earn a D will demonstrate mastery of at least half of the basic competencies and show that they can sometimes, with guidance, draw on multiple competencies to solve new problems.
- Students who earn a F will be those who do not show mastery of at least half of the basic competencies, who disengage from the course, or who violate the academic integrity policy.

A plus on the grade can be earned by showing mastery of at least one third of the competencies for the next higher letter, and a minus on the next letter can be earned by demonstrating mastery of at least two thirds of those competencies. For example, a student who has mastered all of the basic competencies and 70% of the intermediate competencies will be given a B-. As an exception, an A+ can be earned by fulfilling the requirements for an A and also demonstrating mastery of a concept beyond the course's usual scope as negotiated by the student and instructor.

The target competencies for this course are as follows, each listed along with the grade level that it counts towards; basic competencies are marked (C), intermediate competencies are indicated by (B), and advanced competencies are signified by (A).

Programming Concepts

Procedural Programming

* Expressions (C)
* Assignments (C)
* Sequences of Assignments (C)
* Function Definitions (C)
* Conditionals (C)
* Loops (C)
* Single Recursion (B)
* Multiple Recursion (A)
* Nesting (B)
* Higher-Order Functions (A)

Abstract Data Types

* Tuples (C)
* Lists (C)
* Strings (C)
* Maps/Dictionaries (B)

Object-Oriented Programming

* Fields (C)
* Methods (C)
* Visibility (B)
* Mutability and Immutability (B)
* Polymorphism (A)

Algorithms

* Discrete-Time Simulation (C)
* Continuous-Time Simulation (C)
* Finite-Element Models (B)
* Worklists (C)
* Breadth-First Search (B)
* Depth-First Search (B)
* Best-First Search (B)

Software Engineering Skills

Documentation

* Comments (C)
* Docstrings (B)

Version Control

* Manual Repository Setup (A)
* Forking and Cloning (C)
* Staging and Committing (C)
* Pushing (C)
* Pulling (B)
* Conflict Resolution (B)
* Peer review/Pull Requests (A)

Verification Techniques

* System Tests (C)
* Unit Tests (B)
* Coverage Metrics (C)
* Code Inspections (B)

Programming concept competencies will be considered demonstrated when a student passes two associated assessments: a lab to be completed with an assigned partner and either a ten-minute open-book open-note quiz or an untimed homework to be completed individually. Software engineering skills will be considered demonstrated when students show their appropriate use on a programming concept homework (or other task negotiated with the instructor).

All assessments can be reattempted without penalty, though the instructor may assign a student a different variation of the assessment for quiz and homework reattempts.