CMPS 203 Programming Languages, Final Syllabus and Course Information April 12, 2006

Instructor: Allen Van Gelder, 355-E2, 459-4611, avg@cs
Office Hours: Mon., Wed. 3:30–5:00, and appt./drop in

Errata on the internet.

Optional: Stansifer, Programming Languages (1993, 2nd ed.)
Gehani, Narain, C: An Advanced Introduction, ANSI C Ed., 1989(?).

Papers: To be announced and handed out.

Web Page: http://www.cse.ucsc.edu/classes/cmps203/Spring06.
Course Directory: /cse/classes/cmps203/Spring06/
Newsgroup: rn -q ucsc.class.cmps203 (-q is important! Disable threads somehow.)

Course Work: Students will be evaluated on written problem sets from the text, mini-projects assigned to explore various aspects of programming languages, and class participation, which may include discussing written assignments and presenting (informally) some original aspects of mini-projects in class to inform the rest of us about something interesting that you discovered. Each student will also make a “research presentation” of 10–15 minutes with a few slides on some aspect of the course that the student studied in greater depth.

Description: This course is a graduate level introduction to the analysis of programming languages. It contains some overlap with the undergraduate upper-division course CMPS 112, but it puts more emphasis on the issues of correctness and verification. These are studied from a practical point of view. It is expected that the student already has programming experience and is familiar with elementary computational models (finite automata, regular expressions, context free grammars, Turing machines, etc.), discrete mathematics (sets, functions, graphs, discrete probability, proof by (strong) induction, elementary combinatorics, etc.), and elementary calculus. CMPS130 is recommended as a preparatory course for students needing more theory background. CMPS101 and Math 100 are additional preparatory courses that some students may wish to take.

Policy on Academic Dishonesty: Any instance of academic dishonesty is grounds for failing the course, regardless of performance in other aspects of the course. Presenting work as your own when you did not actually do it is dishonest (academically, professionally and socially dishonest) and is called plagiarism. Always attribute (give credit for) anything done by someone else, and then you cannot be guilty of plagiarism. Normally it is not necessary to attribute materials provided by the instructor for the class, or contained in assigned reading for the class, when you use them in class work — we already know about that; however, you would attribute them if you used them outside the context of the class.

Permitting another student to copy your work is also academic dishonesty, except for a group project. Students are expected to exercise reasonable caution that their own work is not copied improperly by another student. NOTE: Putting readable files in the Project directory and reading other students’ files in that directory is OK for this class.

Syllabus: The following lists the topics I “expect” to cover and the approximate order.

Informal computational models and history (read ch. 1 and 2): evolution from automata to block-structured to functional.

Verification with ESCJAVA2 (read Axiomatic Semantics in ch. 13.4, 13.5 and do exercises 13.23–13.28; experiment with proving quicksort is correct using ESCJAVA2; where applicable, use ESCJAVA2 to do the exercises)

Logic programming (read ch. 12, especially 12.1)

Functional programming (read ch. 11)

Abstract Data Types and Object-Oriented Programming (read ch. 9 and 10)