Course Information

Course structure for Spring 2002

The course is an informal survey of programming languages, focusing on programming languages concepts such as binding, evaluation, and types, exhibited on representative languages from the functional, logic, and object-oriented programming paradigms.

The goal is to establish a working knowledge of several languages and to practise the craft of programming in different paradigms. The course will be fast-paced, and will involve *extensive programming assignments*. Previous acquaintance with the procedural and functional paradigm is assumed, as described below.

Assignments and grading

Several (approx. 6) short in-class quizzes (10%) , mid-term exam (15%), final exam (covering the entire course) (25%), about 10 written homework assignments (50%). You are expected to attend class regularly, read the assigned reading before class, and participate in class discussion.

Attendance (same rules as in CS 257)

Your attendance at lectures is mandatory. Unfortunately, it is not practical for me to take attendance before each lecture. Consequently, to encourage attendance, quizzes will be unannounced and in-class.

Prerequisites in detail

Experience with developing substantial applications in functional and imperative (especially object-oriented) programming languages is recommended. UNM CS courses CS 257 - *Nonimperative Programming* and CS 351 - *Design of Large Programs* provide appropriate background in Scheme and C++, respectively. Note that CS 257 is an official prerequisite. If you have not used Scheme before (or another functional language such as ML or Haskell), you must take CS 257 before enrolling in CS 451.

Lectures

Mondays & Wednesdays, 4:00 - 5:15, in Mitchell Hall 111.

Instructor

Darko Stefanovic, office FEC 345C, phone 2776561, email darko@cs.unm.edu — office hours Mondays 3:00-3:50, or by appointment.

Teaching assistant

Mykola Dudar, office FEC 301A, phone 2773394, email dmykola@cs.unm.edu — office hours Wednesdays 2:00-3:50, Thursdays 3:00-5:00, or by appointment.

Textbooks

(The bookstore has ordered the titles marked with *.)

Required reading

* Mark Guzdial: *Object-Oriented Design with Multimedia Applications*, Prentice Hall, 2001, ISBN 0-13-028028-3.

* Laurence C. Paulson: *ML for the Working Programmer, 2nd edition*, Cambridge University Press, 1996, ISBN 0-521-56543-X.

Optional reading

Michael R. Hansen and Hans Rischel: *Programming Using SML*, Addison-Wesley, 1999, ISBN 0-201-39820-6.

Jeffrey D. Ullman: *Elements of ML Programming, ML97 Edition*, Prentice Hall, 2000, ISBN 0-13-790387-1.

Ivan Bratko: *Prolog Programming for Artificial Intelligence, 3rd edition*, Addison-Wesley, 2001, ISBN 0-201-40375-7.

Richard Bird: *Introduction to Functional Programming using Haskell*, Prentice Hall, 1998, ISBN 0-13-484436-0.

Matthias Felleisen and Daniel P. Friedman: *A Little Java, a Few Patterns*, MIT Press, 1998, ISBN 0262561158

Paul Hudak: *The Haskell School of Expression*, Cambridge University Press, 2000, ISBN 0-521-64408-9.

W. F. Clocksin and C. S. Mellish: *Programming in Prolog, 4th edition*, Springer Verlag, 1994, ISBN 3540583505.

Krzysztof R. Apt: *From Logic Programming to Prolog*, Prentice Hall, 1997, ISBN 0-13-230368-X.

Richard Bird and Oege de Moor: *Algebra of Programming*, Prentice Hall, 1997, ISBN 0-13-507245-X.

H. P. Barendregt: *The Lambda Calculus: Its Syntax and Semantics, revised edition*, Elsevier North-Holland, 1984, ISBN 0-444-87508-5.

David A. Watt: *Programming Language Concepts and Paradigms*, Prentice Hall, 1990, ISBN 0-13-728874-3.

Michael J. C. Gordon: *Programming Language Theory and its Implementation*, Prentice Hall, 1988, ISBN 0-13-7304170-X.

Anthony J. Field and Peter G. Harrison: *Functional Programming*, Addison-Wesley, 1989, ISBN 0-201-19249-7.

Homework and programming assignment hand-in policy

This course covers a lot of material and being late with assignments would hamper your ability to learn the next section of the course. Therefore, homework assignments are due on the date assigned, no extensions will be granted, and no credit will be given for late homework. There are very many different formats of text files, very many ways to package them for email, and very many students in the class. Therefore, it is not possible for me to handle electronic submissions. Hard copy solutions must be handed in: either in class on the due date., or during office hours on the due date.

Programming questions on homework assignments

When an assignment asks you to write a program, that means that you must design a program, type it into a computer, compile (depending on the language at hand), and run, and submit a listing of the program and its output. The textual layout of the program must be logically sound and aesthetically pleasing. The names used must be descriptive. Code comments and additional documentation must accompany non-trivial programs, and must provide informal argumentation that the program satisfies the specification. (See below.) Occasionally you may be asked to provide a formal correctness proof of a program—include that proof as part of program documentation.

Cooperation and cheating

Feel free to *discuss* homework assignments with classmates and the instructor. However, *do not look at or copy another's solution*. If a problem appears too difficult, or you lack the background to solve it, you are expected to talk to the instructor promptly. Once you have the background necessary to solve a problem, you must provide your own solution. Exchanging homework solutions is cheating and will be reported to the University administration; students involved may not be permitted to continue in the class. You are responsible for exercising due diligence in protecting your homework files from unauthorized access. In case two or more students present essentially similar homework, all involved students will be reported to the University administration. Occasionally we will reuse problems that have been used in previous courses. Searching on the web for verbatim solutions is not the best application of your time, and you will receive no credit for it.

Lecture Plan Overview (subject to change)

- Organizational meeting; Squeak (1 lecture)
- Scheme review (1 lecture; 1 homework)
- Squeak (4 lectures; 1-2 homeworks)
- Prolog (5 lectures; 2 homeworks)
- Mid-term (1 class period)
- ML (10 lectures (4 + 4 + 2); 2 homeworks)
- Advanced topics (8 lectures; 1 homework)
- Final exam