Syllabus for ICS 313: Programming Language Theory

[Note: This syllabus is based in large part on syllabi for previous versions of this course, particularly the version taught by Prof. Nancy Reed.]

Instructor:	Kim Binsted
Email:	binsted@hawaii.edu
Phone:	808 398 1300 (email strongly preferred)
Office:	POST 303D
Office hours:	Tuesday/Thursday 12pm-1pm, or by appointment
TA:	BJ Tix
TA email:	bjavery@hawaii.edu
TA office:	POST303-1
TA office hours:	Tuesday/Thursday 12:45-2:45pm, or by appointment
Class page:	Laulima

Introduction

This course introduces Unix and Emacs for software development, logical representation and reasoning, and the history of computing and programming languages. Programming language concepts include syntax and semantics; names, scopes, and bindings; control structures and control abstraction; and data types and data abstractions. Programming paradigms learned include functional programming with Common Lisp and/or Scheme, and Logic programming with Prolog.

Learning Objectives/Outcomes

By the end of the course, students should have:

- An ability to use sound development principles to implement computer-based and software systems of varying complexity, and to evaluate such systems. [SO#3]
- An ability to use current techniques, skills, and tools necessary for computing practice. [SO#8]
- An understanding of the differences among programming languages and paradigms, and how these differences impact the development and maintenance of programs.
- An ability to program in different programming paradigms/styles, including functional and logic languages (e.g. Common Lisp, Scheme and Prolog).
- An ability to represent and solve problems at an abstract level before coding them in a particular language.
- An ability to understand new programming language concepts, and assess their usefulness in solving problems.
- An ability to choose the best programming language(s) for a project and justify that choice with well-reasoned arguments based on language characteristics and the development environment.

Textbook and other resources

Required:

- Programming Language Pragmatics, third edition. Michael L. Scott, Morgan Kaufmann Pub., April, 2009 (paperback), ISBN-10: 0123745144, ISBN-13: 978-0123745149. Note: we will not use the CD from the book. Any edition will contain most, if not all, of the relevant material, but the page references might be different.
- Land of Lisp, Learn to Program in Lisp, One Game at a Time! by Conrad Barski, M.D. No Starch Press. October 2010, 504 pp. (paperback) ISBN: 978-1-59327-281-4.

Optional Reference Material:

- Unix/Emacs:
  - Just Enough Unix, any edition, Paul K. Andersen McGraw Hill (or comparable text). Recommended unless you are already proficient with Unix and Emacs. Note: ITS (on campus) has free reference material available on the "flavor" of Unix installed on UHUnix.
- Lisp:
  - Practical Common Lisp, Peter Seibel, Apress, 2005, ISBN: 1590592395.
  - ANSI Common Lisp, Paul Graham, Prentice Hall, 1995, ISBN: 0-133708756
  - Common Lisp, A gentle introduction to symbolic computation, Touretzky. (out of print, PDF available on Laulima under “Resources”).
- Prolog:
  - Thinking as Computation: A First Course by Hector J. Levesque, 978-0262016995, The MIT Press, (Jan 6, 2012). Strongly recommended unless you already have Prolog experience. Slides and sample programs from the textbook are available from the author and on Laulima.
  - Learn Prolog Now http://www.learnprolognow.org/ Programming in Prolog, 4th edition, Clocksin & Mellish, Springer-Verlag, ISBN 3-540-58350-5.

Assignments and Exams

Grades in this course are based on performance in exams and assignments as follows: 6-8 written and/or programming assignments (25%), one programming project plus demonstration (15%), two written midterm examinations (15% each), and a final examination (30%).

For all assignments, you have the option of submitting it at least a week in advance of the deadline for instructor feedback. See “Schedule” on Laulima for deadlines. All assignments should be submitted via Laulima by 11:55pm on the due date, USUALLY A FRIDAY. After that (even by a minute!), assignments will lose 10% per day for each day late, up to five. After five days, they will not be accepted.

The final will be held according to the campus-wide schedule (http://manoa.hawaii.edu/undergrad/schedule/final-exams/fall/), which says it will be the

Thursday of exam week 9:45am-11:45am (subject to change, so please double-check before exam week). If you do not expect to be on campus for this time slot, please *withdraw from this course now.* Only serious, documented, unexpected events (i.e. not “But I have a plane ticket!”) for missing the final will be grounds for alternative arrangements.

Submission Procedure

- Use Laulima's Assignment tool (do NOT use email unless specifically stated). If you have problems submitting in the assignment area, submit to your Dropbox folder on Laulima, then notify the TA. If you are not yet registered for the course, contact the TA before the assignment is due. It’s a good idea to submit early, in case there are technical issues.
- All Lisp files submitted must contain a header at the top with identifying information in it (see “header.lisp” under “resources” on Laulima for an example). At a minimum, you need the name(s) of all the author(s) of the code, the course ID, assignment number, and the current date. For multi-file programs and large files with many related functions, add a short description of your solution in the header of your main file.
- Formatting guidelines are shown in FormattingLispCode.pdf (under “resources” on Laulima). Include a documentation string in all functions and declarations. Indentation should be 2 or 3 spaces for each new level of indentation '('. Vertically align/indent code at the same level. Emacs will format/reformat your code in an appropriate style.
- Turn in your SOURCE code files - all the functions that make up your program, including comments. All programming code and text must be consistently formatted (indentation, etc) and must also be documented - header blocks for functions, side comments for variables and inside code. Use a good text editor (e.g. Emacs) that can automatically indent your code and do syntax checking for you, besides the usual editing functions.
- Create and test all programs on a complete (representative) sample of test data! Designing good test data (for programs and functions) is strongly correlated with your ability to create good, robust programs. If you don't test it, it probably won't work. You may create additional files/documents with design and testing information.
- Turn in output (one or more plain text files) demonstrating that your programs work on your tests: script (Unix) OR dribble (lisp function) OR saved buffer (Emacs) OR screen copy/paste (SSH). Show results on all cases provided as well as tests you have created. Not considering an important type of test will likely lose points.
- Programs must load (or compile) and run on the platform specified by your instructor. If they fail to load or compile and execute properly (on at least some input data), you will receive no credit.
- Collaboration on individual assignments is not permitted. Group assignments, if any, will be explicitly identified in the assignment. One copy of the code should

be submitted for each group. Each group member must also submit a report describing their own work on the project.

- Keep a copy of your submission. Make sure than your name and assignment number are in all files submitted. They should be in comments in source code files, so that the files still run/compile correctly.
- You must submit a reasonable attempt at solving each and every assignment in order to pass the course, i.e. to receive a grade of C or better.

Grading Breakdown

Assignments (6-8)	25%
Programming Project & Demo	15%
Midterm (2 x 15%)	30%
Final	30%

97-100%	A+
94-96%	A
90-93%	A-
87-89%	B+
84-86%	B
80-83%	B-
77-79%	C+
74-76%	C
70-73%	C-
67-69%	D+
64-66%	D
60-63%	D-
< 60%	F

I do not grade on a curve. If the ranges above are changed, they will only be adjusted downwards (e.g. if you have 90%, you will at least receive an A-).

Academic Honesty

Plagiarism and cheating are not tolerated in this course. If a student is caught cheating, plagiarizing or helping others to cheat, s/he will fail the course, and further disciplinary action may be taken. Please be sure to review the definition of plagiarism, and guidelines on how to avoid it, here:

http://www.cte.hawaii.edu/plagiarism/plagiarism_1.html

Please also review the Association for Computing Machinery's code of ethics:

http://www.acm.org/about/code-of-ethics/

KOKUA Program

If you feel you need accommodations due to special circumstances, please contact the KOKUA Program (V/T) at 956-7511 or 956-7612 in room 013 of the QLCSS, and/or speak with me privately to discuss your specific needs. I am happy to work with you and the KOKUA Program to provide the support you need to succeed in this course.

Changes to the Syllabus

The instructor reserves the right to change the syllabus at any time. [Congratulations, you found the hidden bonus question! In your response to Assignment 1, please say in one sentence why you’re taking this course, and get five extra points.] If any changes are made, the revised syllabus will be posted on Laulima, and students will be informed by email. Please be sure to inform the instructor and TA if your contact information changes.