Syllabus for CSC 3/458: Symbolic Programming, Winter 2018

Overview

This course is a hands-on introduction to symbolic programming and to Common Lisp. Lisp is the language of choice for developers who want to develop interesting software, to do it quickly, and to enjoy doing it. In this course, we will explore the following topics: LISP syntax and semantics, Conditions, Games, I/O, Lambda, Functional programming, Data types, Generic programming, and Lazy programming.

Learning Outcomes

By the end of this course you should be able to:

- describe the power of the symbolic programming approach,
- apply a variety of symbolic programming techniques,
- recognize and describe the types of applications that these techniques are especially good for, and
- develop and implement an approach to solving a given problem using symbolic programming techniques.

Prerequisites

CSC 301 or CSC 383 or CSC 393 or CSC 403

Textbooks

Required:

- Land of Lisp by Conrad Barski. No Starch Press. ISBN 13: 9781593272814
  Also available via Safari books online through the DePaul Library

Recommended:

- ANSI Common Lisp by Paul Graham. Prentice Hall. ISBN: 0133708756

Attendance

Class attendance is strongly encouraged, but not mandatory. However, if you are absent from class you are responsible for understanding the material and for finding out about any announcements made in that class.

Class Plan

The following class plan is tentative and subject to change as the course progresses.

- **Class 1:** (1/8) Course overview. Introduction to Symbolic Processing.
- **Class 2:** (1/15) No class due to MLK day Holiday. BUT, there are readings and a Warm-up due before classtime. Topics: Loading a LISP, First program, Syntax of LISP
- **Class 3:** (1/22) Conditions, A text game engine

- **Class 4:** (1/29) I/O, The Lambda function, Advanced lists
- **Class 5:** (2/5) Grand Theft Wumpus (with sets), Looping with LOOP, Debugging and Style
- **Class 6:** (2/12) Advanced datatypes, Generic programs, FORMAT, Streams
- **Class 7:** (2/19) Functional program, Applied to Dice of Doom game
- **Class 8:** (2/26) Macros, Domain-specific languages
- **Class 9:** (3/5) Lazy programming, Making Dice of Doom graphical and web-based, Making it more fun
- **Class 10:** (3/11) Project Presentations
- **Class 11:** (3/18) Project comments due

Instructor Information

Email	peterh@cdm.depaul.edu
Home Page	http://reed.cs.depaul.edu/peterh/
Phone	312-362-5736
Office Hours	Monday 3:30-5:00PM at my office, Thursday 4-5:30PM at LPC or by arrangement
Address	CDM Center 717 DePaul University 243 South Wabash Avenue Chicago, IL 60604-2301 USA

Assessment

Your final grade will be based on:

Item	Pts
Weekly warm-ups*	20
In-class exercises	20
3 programming assignments	75
Final programming project	50
Final project report	15
Project presentation	10
Comments on other presentations	10
Total	**200**

The grading scale will be:

Pct	Pts	Grade
93.3	186.6	A
90	180	A-
86.6	173.2	B+
83.3	166.6	B
80	160	B-
76.6	153.2	C+
73.3	146.6	C

Pct	Pts	Grade
70	140	C-
66.6	133.2	D+
60	120	D
< 60	< 120	F

All homework assignments will count towards the final grade. Some parts of the homeworks will be labeled as extra credit. These parts are **required** for graduate students.

Weekly Warm-ups

I use a "Just In-Time Teaching" methodology which is intended to help students by lining up what's done in and outside of class. So each week (after the first), you will be given some "warm-up exercises" to help you come to grips with what you've read, and to help me see what you're getting and what you're not. (That way the class time can be tailored to your needs.) For this class, the warm-ups will focus on the "tutorial" aspects of the code in the book, to ensure that you've run it, and that you understand what it means. The warm-ups must be completed before noon (i.e. by 11:59 AM) on the day of the class. (This applies to both in-class and online students.) The lowest score will be dropped. These will be worth a maximum of 5 point each.

Weekly "in-class" work

In each class, we will spend some time discussing issues related to the material, and working on small exercises. In-class students will be able to work with other students on most of these. Online students students are *highly encouraged* to join "virtual study groups" with other online students to facilitate this. Educational research shows that one of the strongest predictors of deep learning in classes is participation in study groups. I will do whatever setup is necessary in D2L to support this. Online student can submit their "in-class" work via D2L discussion forums. These are also worth a maximum of 5 points per week.

Programming assignments

The programming assignments for this course will involve applying what you have learned from the readings and the class, transferring that knowledge to different problems.

The final project will be on a topic of your choice (with instructor approval) and include a presentation to the class that describes your project at the last class meeting. This will be described in more detail later.

Late submission policy

The programming assignments are due via D2L at the time indicated on the course homepage. Because we will be discussing the answers to the programming assignments in class (the day after the assignments are due), no late assignments will be accepted. If you are not done by the deadline, submit what you have for partial credit, and indicate what works and what does not.

On Plagiarism

You are encouraged to discuss all homeworks and projects with your classmates. You are, however, required to complete them on your own. In particular, this means that you are not allowed to "cut and paste" code from anywhere else for the programming assignments.

On using code libraries for the final project

Students often want to create a big, impressive project by using code libraries. This causes two problems:

- The student usually spends an inordinate amount of time trying to figure out how to get the library to work.
- I can not tell how much the student learned about symbolic programming from this.

So I discourage the use of these libraries. I am much more impressed by a modest project that you complete yourself, than by an ambitious project that relies primarily on someone else's code.

If you do use one or more code libraries, you need to be very clear about where it came from and distinguish what you did yourself from what you got from somewhere else.

School policies on instructor evaluation, email, plagiarism and incompletes.