Course Syllabus for CS 335
Programming Languages
Spring 2011

Basic Course Information:

Instructor:	Sharon Tuttle		
Lecture time and location:	Tuesday, Thursday	12:30 – 1:50 pm	BSS 308
Instructor's office:	BSS 322		
Instructor's e-mail:	st10@humboldt.edu or sharon.tuttle@humboldt.edu or smtuttle@humboldt.edu		
Instructor's office phone:	(707) 826-3381		
Instructor's office hours:	Monday, Wednesday	3:00 – 4:30 pm	
	Tuesday	2:15 – 3:15 pm	
	Thursday	10:00 - 11:00 am	
	or by appointment		
Course public web page:	follow link from: http://users.humboldt.edu/smtuttle/ or follow link from course Moodle site		

Course Description:

[adapted from the HSU catalog] An in-depth treatment of programming languages, including their history, data types, data control, sequence control, run-time storage, language translation, and semantics. Programming models will include procedural, functional, logic, and object-oriented programming.

CS 335 studies "high-level" general-purpose programming languages, with the emphasis on design and semantics. The primary purpose of this course, and the one which will be addressed in the lectures, is to get an overview of the characteristics of programming languages in general. Thus, we will look at such topics as how languages are used, what concepts play an important role in their definitions, how they can help in creating "correct" programs, and other aspects of languages that are common to all of them.

Course Objectives:

After successfully completing this course, students should be able to: *

- summarize the evolution of programming languages illustrating how this history has led to the programming models available today;
- explain the tradeoffs and issues involved in the design of various language features;
- identify distinguishing characteristics for each of the programming models covered in this course;

---

* Some of these are adapted from the ACM Computer Science Curriculum 2001, available from link at: http://www.acm.org/education/curricula-recommendations

- evaluate the tradeoffs between the different programming models, considering issues such as space efficiency, time efficiency (of both the computer and the programmer), safety, and power of expression;
- solve problems using the functional, object-oriented, and logic programming models;
- explain and answer questions about specific languages in which one can illustrate different programming models, including questions about relevant concepts and major features;
- explain the benefits of intermediate languages in the compilation process;
- compare and contrast compiled and interpreted execution models, outlining the relative merits of each;
- describe the phases of program translation from source code to executable code and the files produced by these phases;
- explain the differences between machine-dependent and machine-independent translation and where these differences are evident in the translation process;
- identify and describe the properties of a variable such as its associated address, value, scope, persistence, and size;
- discuss type incompatibility;
- demonstrate different forms of binding, visibility, scoping, and lifetime management;
- defend the importance of types and type-checking in providing abstraction and safety;
- evaluate tradeoffs in lifetime management (reference counting vs. garbage collection);
- demonstrate the difference between different parameter passing mechanisms;
- explain the role of different abstraction mechanisms in the creation of user-defined facilities;

Even if you do not become a programmer, the ideas of the functional programming model (functional abstraction, infinite data structures, continuations, referential transparency) have important applications in all areas of computer science and in many other contexts such as mathematics and engineering. Similar comments hold true for the logic and object-oriented programming models. For example, the idea of data abstraction is certainly a key concept in software engineering and even in contemporary mathematics (category theory).

Understanding the strengths and weaknesses of the various programming models and language features is important in applying them to solve problems. Problems in the real world are not labeled with the programming model that should be used to solve them, so the choice (or choices!) of programming model can be important. In programming language and software engineering research, understanding the strengths and weaknesses of the existing programming models and language features is important for designing better ways to program.

A second purpose of this course, dealt with in part in programming assignments, is to get some exposure to several real programming languages different from those you have most-used thus far. You will likely program in at least the languages of Scheme, Prolog, and Smalltalk; you may also be programming in additional languages as well. Much of this programming will be done in the UNIX/Linux environment, although cross-platform free versions of these languages are also available.

Course Prerequisites:

CS 233 with a grade of C or better, or instructor consent

Required Course Text, Materials, etc.:

- Bruce J. MacLennan, "Principles of Programming Languages: Design, Evaluation, and Implementation", Third Edition, 1999, Oxford University Press, ISBN # 0-19-511306-3
- Additional required readings will be made available either on-line, or via resources available through the HSU

Library such as the ACM Digital Library and Safari TechBooks Online.

- **Either**:
  - Turning Point RF Response Clicker, available at the campus bookstore,
  - **or** a Responseware license, available on-line (address to be provided).

Course Software:

Throughout the semester, you will be making some use of the UNIX operating system on nrs-labs.humboldt.edu. Note that you may access nrs-labs.humboldt.edu by using `ssh` (secure shell) and `sftp` (secure ftp); `ssh` may be downloaded for free from:

https://experts.humboldt.edu/ditss/download/ or
http://www.humboldt.edu/its/software

...although in Fall 2008 some students had better luck downloading it from:

http://www.colorado.edu/its/docs/authenticate/printouts/win_ssh.html

...which also includes a nice illustrated tutorial for this Windows implementation of `ssh` and `sftp`.

In addition to programming within UNIX/Linux, we will be using a number of integrated development environments for different languages. Many of these happen to have free cross-platform versions available, which I will let you know how to download as we need them.

Some software that I think we will be using, at this point:

- Racket - available from: http://racket-lang.org/
- SWI-Prolog - available from: http://www.swi-prolog.org/
- Squeak - available from: http://www.squeak.org

We *might* also make some use of Python -- it is available from http://www.python.org

Clickers:

We will be using Turning Technologies student response clickers (or, for those who prefer, Responseware on one's cell phone or laptop) in lecture. There is significant literature indicating that using clickers may increase student engagement and success in learning.

Students purchase this clicker or buy a Responseware license; purchased clickers can be returned at the end of the semester for a partial refund of the purchase price. Students with clickers register them once, at the beginning of the semester, by entering the large number (consisting of 6 characters/digits) on the back of the clicker at a special address that I will provide, and then bring them to every class meeting. Students using Responseware purchase the license, and then sign into Responseware at the beginning of each lecture, using a special code that I will project at the beginning of each lecture.

These clickers will be used for in-lecture questions, which will be interspersed within the lecture. The response system will record the overall class response percentages as well as keep track of individual answers. Students will receive **2 points** for a correct answer, **1 point** for an incorrect answer, and **0 points** for no answer, but with a maximum semester clicker-questions grade of **120**. Thus you will be rewarded for regular attendance and participation. If you miss a class session, you miss that day's clicker questions and cannot make them up. However, there will be at least **65** questions asked over the course of the semester, so there is opportunity for extra credit (up to a maximum clicker grade of **120**) (or to make up for a day that you are out due to illness, although note that you are still responsible for finding out what you missed on such days).

If you forget your clicker for a class meeting, then **up to 5 times** you may still receive some clicker credit, **minus a 2-point penalty**, by e-mailing me your clicker answers for that day, **by midnight on that day**, using a

Subject: line of: `Subject: CS 335 Clicker Answers for <date>`. Later e-mails, or e-mails without the proper Subject: line, will not be accepted for credit.

The idea is that the clicker questions will help you to see if you are starting to understand concepts being discussed; sometimes they will also provide review of concepts discussed previously. Clicker questions are typically quite different from exam questions (since clicker questions are typically multiple-choice questions, while exam questions will rarely be multiple-choice). They still enable you to get some immediate feedback regarding whether you are grasping course concepts, whether you need to pay more attention to course discussions and/or readings, etc. They may even help me to know what concepts might need more explanation in-class.

I hope to run tests of the system during Thursday's lecture during the first week of the semester, and hope to begin asking questions that "count" during the second week of the semester. Therefore, you must purchase your clicker and register it (or purchase a Responseware license) as soon as possible. If there is an issue with this (for example, if the bookstore runs out of clickers), contact me immediately.

Finally, please note that use of another CS 335 student's clicker, or having someone else use your clicker in CS 335 lecture -- that is, pretending that a student is in class who actually is not -- is considered to be cheating, with the same policies applying as would be the case if you turned in someone else's work as your own or permitted someone else to copy your work. Please ASK ME if you are not sure what I mean by this.

Grading Breakdown:

If you are a Computer Science (CS) major, it is important that you note that you must earn **at least a C in CS 335** for this course to count towards your major and to be able to take courses that require CS 335 as a prerequisite. If you are a Computer Information Systems (CIS) major, it is important that you note that you must earn **at least a C in CS 335** for it to be able to count as a CIS major elective.

Your semester grade will be determined by the percentage of points that you earn, **subject to some minimum requirements**. Here are the grade percentages, followed by those minimum requirements:

Homework assignments:		35%	
Clicker questions:		15%	
Exams:	**Exam #1:**	15%	
	Exam #2:	15%	
	Final Exam:	20%	Thursday, May 12, 12:40 - 2:30 pm, BSS 308

Grade Requirements:

1. To earn a grade of **C or better** in this course, the following three requirements must **all** be met:
   - your overall semester average must **equal or exceed 72.5%** - this is to show a reasonable level of overall mastery of the course material.
   - the **average** of your Exam #1, Exam #2, and Final Exam grades must **equal or exceed 60%** - this is to show that you understand at least a minimal reasonable level of the most important course concepts.
   - the **average** of your Homework assignments must **equal or exceed 60%** - because there is some programming in this course (particularly using different programming models), but programming acumen is not tested as effectively on exams, this is to show at least a minimal level of programming competence and experience in addition to course concept mastery. Also, past experience has shown that students who do not put a solid effort into the course homework assignments do not do well on the course exams.
2. If **all three** requirements above are **not** met, then your semester grade will be **either C-** or the letter grade computed according to the mapping given below, **whichever is lower**.

   (That is, if a student had an overall semester average of 74% but a Homeworks average of 55%, that student

would receive a **C-** for his/her semester grade; if a student had a Homeworks average of 61% and an Exams average of 71%, but an overall semester average of 65%. then that student would receive a **D** for his/her semester grade. You are expected to ASK ME if this aspect of the grading policy is not clear to you.)

3. Including the three requirements noted above, your semester grade will be computed according to the mapping given below:

Overall Percentage (based on the given weights)	**Exams Average**	**Homework Average**	**Letter Grade**
>= 93	>= 60	>= 60	A
>= 90 and < 93	>= 60	>= 60	A-
>= 87 and < 90	>= 60	>= 60	B+
>= 83 and < 87	>= 60	>= 60	B
>= 80 and < 83	>= 60	>= 60	B-
>= 77 and < 80	>= 60	>= 60	C+
>= 73 and < 77	>= 60	>= 60	C
>= 73	< 60	any	C-
>= 73	any	< 60	C-
>= 70 and < 73	any	any	C-
>= 67 and < 70	any	any	D+
>= 60 and < 67	any	any	D
< 60	any	any	F

Final Exam:

Again, the Final Exam for this course is scheduled for **Thursday, May 12, 12:40 – 2:30 pm**, in **BSS 308** (unless I announce otherwise). Note this time and date BEFORE making your end-of-semester travel plans.

Students with Disabilities:

Persons who wish to request disability-related accommodations should contact the **Student Disability Resource Center** in the Learning Commons of the Lower Library, **826-4678 (voice)** or **826-5392 (TDD)**. You can reach the Student Disability Resource Center's web site at:

http://www.humboldt.edu/disability/

Please note that some accommodations may take up to several weeks to arrange. If you are eligible for such accommodations, please contact me as soon as possible to discuss them.

Add/Drop Policy:

Students are responsible for knowing the University policy, procedures, and schedule for dropping or adding classes. You can find these on the web at:

http://www.humboldt.edu/registrar/students/regulations/schedadjust.html

You can find the University policies for repeating classes at:

http://www.humboldt.edu/registrar/students/regulations/repeat.html

Note that the CSU (and thus HSU) policies on withdrawing from and repeating courses changed as of Fall 2009:

- Students may withdraw from no more than 18 semester-units after the first four weeks of instruction; that is, students may withdraw from no more than 18 semester-units between census and the final 20% of instruction, and only then with a serious and compelling reason.
- Students may repeat courses only if they earned grades lower than a C.
- Students may repeat up to 16 semester-units with grade forgiveness.
- Students may repeat up to an additional 12 semester-units with grades averaged.

Be careful – as of Fall 2009, HSU is being much more strict about what constitutes a "serious and compelling reason".

The census date for Spring 2011 (before which you can drop without a W, and without it counting toward your 18 semester-units drop limit) is: **5:00 pm on Monday, February 14th**.

The last date for Spring 2011 to drop with a W on your transcript, with a serious and compelling reason, and subject to the 18 semester-unit drop limit, is: **Friday, April 1st**.

If you do drop the course, note that it is **your responsibility** to complete and submit the appropriate paperwork. As noted in the University policies for dropping courses, "As a matter of university policy, **the instructor in the course may not drop on behalf of the student**." (This may change in Fall 2011, but it is still the case for Spring 2011!)

Incompletes:

Incompletes are rarely given and only in the case of a true emergency. They certainly are not appropriate for students who find they have fallen behind on assignments, missed a test, or taken on too much academic, work, or family responsibilities. For these situations, dropping the course would be appropriate (**if** that is still possible according to the University policies for dropping courses).

Time Expectations:

Remember the general rule of thumb for college-level courses --- to be successful in a course, you should plan to spend at least 3 hours outside of class for each 1 hour of college course credit. That implies an estimate of at least 9 hours a week spent outside of class for this 3-credit course.

However, you should be warned that:

- This is a junior-level CS major course; it has an correspondingly-rigorous workload.
- Although not every homework in this course will include programming, many of them will. You should know by this point that programming courses can be notorious time eaters -- occasionally, a problem with code will take large amounts of time to locate and fix. Starting early enough so that you have time to ask me questions when you run into problems can help with this. Why spend 4 hours struggling with a frustrating roadblock the night before the homework is due, when you can spend 10 minutes composing an e-mail early in the week, work on other problems while waiting for the answer, and then get a reply that makes everything clearer as soon as you read it?
- You can only get more comfortable with different programming models by practicing them, and it takes some much longer than others to master them. Practicing as much as possible helps. (This means playing around with in-class examples, experimenting to see if something you are curious about really works like you think, and so on.)
- Later concepts are built upon earlier concepts as the course progresses -- if you ask me as soon as you realize that some concept is not clear to you, that can help keep you from falling behind.

- Homework deadlines will not be extended because you waited too late to start or because you did not allocate enough time before the deadline to work on it; likewise, they will not be extended because of hardware or network failures. You need to keep backups of your files at all times, and need to plan your schedule to be able to work on on-campus computers as necessary.
- If you have not completed an assignment by the deadline, your best choice is to submit whatever you have managed to do by then, as partial credit is your friend, to carefully study the posted example solution as soon as it is available, to ask me about anything there that is still unclear, and to get a good early start on the next homework.

Academic Honesty:

Students are responsible for knowing policy regarding academic honesty. For more information, visit:

http://www.humboldt.edu/studentrights/academic_honesty.php

Observe that among the actions that are unacceptable are submitting another's program, code, or file as your own and failing to quote material taken from another person's written work. (Note that copying another student's comments is also unacceptable.)

All course work is to be the work of each student, **individually**, **unless** it is **explicitly** stated otherwise at the beginning of that course work's description. Except for explicit exceptions, this is **not** a group or team programming course. If group work is explicitly permitted for some assignment, the names of all students involved must be included on the work submitted. (For example, if **pair programming** is explicitly specified as being allowed for an assignment, then each pair-programmed file turned in will include both of the names of the students who worked on it as a pair.)

(**Important aside**: pair programming specifically means that two people sit at one computer, with one typing while the other says what to type. Both people are actively involved in the programming process. Pair-programming is **not** two people working at two computers, each doing different parts of the work individually. If pair-programming is ever explicitly permitted, then you are expected to actually pair-program any files you do not complete on your own.)

(If an assignment does explicitly specify that it is acceptable to pair program or work in groups, make sure that you don't get into the situation where you are merely watching someone else learn.)

For homework assignments (that are not explicitly specified as permitting pair-programming), students may discuss general approaches **as long as no one involved in the discussion is writing anything down or typing anything during such discussions**. Students may also help one another in determining causes of program bugs, or in determining the meaning of compiler error messages. However, in general, students may not work together to complete homework assignments, one student should not instruct another in how to write the code for a homework assignment, and **any type of copying or modifying of another person's computer files, OR of providing computer files to another, related to homework assignments is definitely over the line, and never justified**. This applies to copying of documentation and comments as well as to copying of program code.

Note that it is **your** responsibility to ensure that course assignment files are read-protected. If you are careless about this, and someone else copies your work, you will share the penalty. (In particular, be very careful about leaving work on shared network drives in campus labs, or in UNIX/Linux directories that are not read-protected.)

Learning takes hard work; when students turn in others' work as their own, it is a slap in the face to those seriously interested in learning. Not turning in an assignment results in no credit for that assignment, of course, but that is an honest grade. Work that violates the course honesty policy deserves a lower grade than that, and therefore the course policy is that work violating this policy will receive **negative** credit. A person providing a file for copying receives the same **negative** credit as the copier. Repeat offenses will be handled according to University policies.

Asking Questions/Getting Help:

- You are encouraged to ask me questions in class, in office hours, and by e-mail. The most successful students are those who are not afraid to ask questions early and often (I will gently let you know if you are overdoing it), who do the assigned reading, who attend lecture regularly, who start homeworks promptly after they are made available from the course web page, and who practice course concepts as much as possible.
  - It is better to ask a question sooner than later -- for example, it is better to send an e-mail with a specific question as soon as you think of it than it is to wait a day or two until the next class meeting or office hour. If you wait to ask such questions, you may not have time to complete the assignment.
  - It is perfectly reasonable if you send me a question and then end up finding out the answer yourself before you receive my answer; likewise, it is not a problem if you end up sending me several questions in separate e-mails (as you work on different parts of a homework while awaiting earlier answers).
- That said, I am expecting that you will ask **specific** questions – overly vague or broad questions are problematic. (For example, an example of a specific question is, “When I try to run the function: (paste in the function), I receive the following error message: (paste in the error message) Can you point me in the right direction about what is wrong?” An example of an overly vague or broad question is: “Here's my program/function/answer/etc. Is it right?”)
- I try to check my e-mail (`st10@humboldt.edu` or `sharon.tuttle@humboldt.edu` or `smtuttle@humboldt.edu`) about once a day on weekdays, and about once over each weekend. This is another reason to start assignments early, so that you have time to receive a reply to any questions that might arise. Include `CS 335` and a general description of your topic in the Subject: line, both because including this makes it less likely that I'll overlook your question, and because it will make your message stand out if it the spam filter gets confused and puts it in the university spam quarantine.
  - If I have not replied to your e-mail within 24 hours, please re-send it, just in case it did get overlooked somehow.
  - Also, DON'T INCLUDE the word "`password`" in your e-mails to me -- `pwd` is a handy abbreviation to use instead -- because, due to phishing scams, HSU's spam filtering definitely does not like e-mails with that word in it! (Odd, but this was definitely the case in Spring 2010...)

Additional Coursework-Related Policies:

- You should expect to put in a significant amount of time outside of lectures doing the assigned reading, working on homework assignments, and practicing concepts discussed.
- Assigned programming is expected to be done using the software/languages/language versions indicated in the homework handout; "on-paper" homework problems are expected to follow any assumptions/requirements given.
  - Code that does not run on the campus computer(s) and/or the environments specified will not receive credit; remember that it is your responsibility to verify that your code runs on on the specified campus computer(s)/environment(s) for each homework before submitting its code, regardless of where you developed that code.
- Each assignment must be submitted as is specified on its handout to be accepted for credit. This may vary for different assignments. Often, parts of assignments will be submitted using a special tool on nrs-labs.
- Each assignment will be clearly marked with one or more due dates (a single assignment could have multiple parts with multiple due dates).
  - **No assignments will be accepted late. If you wish to receive any credit for an assignment, then you must turn in whatever you have done, even if it is incomplete, by the deadline. Partial credit is usually preferable to no credit.** Note that "the computer/network/etc. going down" is no excuse --- if you

leave an assignment for the last minute and there are technical problems, you still must turn in whatever you have by the deadline. As with any work done on computer, make frequent back-ups of your files!

  - You may submit multiple versions of assignment files before the deadline; I will grade the latest pre-deadline submission unless you inform me otherwise. This is to encourage you to turn assignment parts in early (since you will know that you can always turn in an improved version if further inspiration strikes). You also don't have to worry about forgetting to submit something that has already been submitted.
  - In this course, some parts of some homework assignments will be turned in on-paper; on-paper portions of homework assignments must be given to me, in person, by the appropriate deadline or turned into my mailbox or the department drop-box in the Mathematics/Computing Science department office by that deadline. On-paper portions of homework assignments that I find leaning against my office door after the deadline will **not** be accepted.

- The tool that you will be using to submit some assignment parts results in a file that serves as your "receipt" for having submitted items. You are expected to retain these "receipt" files at least until a grade has been posted to the course Moodle site for that assignment. If there is a system glitch or other hardware/software/network problem, you may be asked to make me a copy of one or more receipt files; if you do not have them, then you will not receive credit for the files involved. These receipt files are for your protection!
- It is nearly impossible to write unambiguous specifications. If you have questions about "what she means", get them resolved very early in the development cycle by **asking**.
- There is more to computer code than simply whether it runs or not...
  - Part of your grade will be determined by how well your work meets the written requirements. Work that you turn in is expected to meet handout specifications precisely; when one eventually works within a team on large projects, following the specifications precisely is vital, and can mean the difference between a working product and one that just sits there.
  - Note that work may be graded on **style** as well as on whether it runs properly and whether it precisely meets the homework specifications and requirements. Discussions on style will be ongoing throughout the semester -- and in this course, we will also be discussing how programming style varies for different languages and different programming models!
  - In this course, use of the appropriate programming model will also be a vital component, and so could be a factor in grading as well.
- Some course work may be graded simply based on whether it has been attempted (the instructor's decision is final as to whether this is the case) -- other course work may be graded for correctness, style, and whether it meets specifications. You will not know in advance which will be the case.

Additional Grading-Related Policies:

- If you would like me to e-mail certain course grades to you during the semester, then you must give me permission in writing on the course information form.
- Clicker questions will be given during most lectures. Because of the ample quantity of clicker questions asked, you can be absent several times from non-exam lectures without direct penalty, for whatever reason (although you are, of course, still responsible for the material covered on those days, and it is your responsibility for determining what that material is).
- Note: **NO** homework grades are dropped; **ALL** homework grades count toward your homework average. Every homework includes important practice of course fundamentals.

Additional Course Policies:

- You are expected to read this syllabus and be prepared to sign a statement that says you have received it, have read it, and understand its contents.
- Tentative exam dates are given in the course schedule below. Make-up exams are only possible by special prior arrangement or because of a valid medical excuse.
- You should monitor your e-mail for course-related messages. The University provides a means for you to specify your preferred e-mail address, so if you wish to receive e-mail into an account other than the one HSU provides, change your preferred e-mail address in both Account Center and Moodle accordingly. Course-related messages from me will include `CS 335` in the Subject: line.
- You are expected to check the public course web page and the course Moodle site regularly --- course handouts, homework assignments, examples from lectures, and possibly more will be posted to the public course web page, and grades will be posted to the course Moodle site. You are expected to monitor your posted grades and let me know about any discrepancies.
- When reading assignments are given, you are expected to prepare (read and study) assigned readings before class and to participate in class discussions. Projected examples will be utilized frequently during discussion. You should understand that there may be material in the reading that will not be discussed in lecture, and material in the lectures that may not be found in the reading. You are responsible for both.
- **Attendance and disruptive behavior:** Students are responsible for knowing policy regarding attendance and disruptive behavior:
  http://www.humboldt.edu/studentrights/attendance_behavior.php
- Regular attendance at lecture sessions is expected. If you should happen to miss a lecture, then you are responsible for finding out what you missed. "I wasn't there that time" is never an acceptable excuse. Lecture notes are not posted, although many of the projected examples will be made available on the public course web site. Clicker questions missed **cannot** be made up later.
- **Late arrival to class:** Please attempt to come to class on time, with your headphones put away and your cell phones turned off. If you must arrive late or leave early, please do so with the least possible distraction to other students. If your late/early habits become disruptive, you may be asked to leave the class permanently.
- **Class disruption:** University policy requires that instructors eliminate disruptions to the educational process. Distractions such as excess talking, ringing cell phones, working on assignments for other classes, inappropriate or distracting laptop/tablet/smartphone/gadget use, demonstrations of affection, packing of books early, loud music leaking from headphones, chronic late arrivals or early departures, excessive comings and goings or other behaviors that disrupt the class are not acceptable. Students indulging in such behaviors will first be warned before being required to leave the class permanently.
- **Emergency Evacuation:** Please review the evacuation plan for the classroom (posted on the orange signs), and review the campus Emergency Preparedness web site at:
  http://www.humboldt.edu/emergencymgmtprogram/campus_emergency_preparedness.php
  for information on campus Emergency Procedures. During an emergency, information regarding campus conditions can be found at **826-INFO** or:
  http://www.humboldt.edu/emergency

Tentative Course Schedule: (subject to change!) (last modified: 03-03-11)

Week 1: January 18, 20

- Topics: Intro to course; Specifying syntax: Formal Languages: Regular Expressions (RE's), Context-Free Grammars (CFG's), Backus-Naur Form (BNF)

Week 2: January 25, 27

- Topics: Specifying syntax: Formal Languages: Regular Expressions (RE's), Context-Free Grammars (CFG's), Backus-Naur Form (BNF) continued; also Derivations and Parse Trees
- Homework 1 out

Week 3: February 1, 3

- Topics: Intro to functional programming model, and intro to Scheme
- Homework 1 due; Homework 2 out

Week 4: February 8, 10

- Topics: Continuing with the functional programming model and Scheme
- Homework 2 due, Homework 3 out

Week 5: February 15, 17

- Topics: Control flow; Review for Exam 1
- Homework 3 due

Week 6: February 22, 24

- Tuesday, February 22: **Exam 1**
- Thursday, February 24: Names, Bindings, and Scopes: the Notion of Binding Time; object storage and lifetime management
- Homework 4 out

Week 7: March 1, 3

- Topics: Intro to logic programming model, and intro to Prolog
- Homework 4 due; Homework 5 out

Week 8: March 8, 10

- **TUESDAY, MARCH 8 - GUEST LECTURER!** You are **expected** to be in class -- you will receive 5 clicker-questions credit for attending and signing the roll sheet
- THURSDAY, MARCH 10 - NO CLASS, instructor at conference -- work on Homework 5
- Homework 5 due; Homework 6 out

Spring Break - March 14-18

Week 9: March 22, 24

- Topics: Continuing with the logic programming model and Prolog;
- Homework 6 due; Homework 7 out

Week 10: March 29

- Tuesday, March 29 - Object lifetimes, storage allocation mechanisms, automatic vs. manual storage reclamation, basics of garbage collection;
- Thursday, March 31 - NO CLASS, Cesar Chavez Holiday
- Homework 7 due

Week 11: April 5, 7

- Tuesday, April 5 - Review for Exam 2
- Thursday, April 7 - **Exam 2**

Week 12: April 12, 14

- Topics: Parameter Passing Modes; taking the object-oriented programming model to an extreme: intro to Smalltalk/Squeak
- Homework 8 out

Week 13: April 19, 21

- Topics: Continuing with the object-oriented programming model taken to the extreme and Smalltalk/Squeak
- Homework 8 due, Homework 9 out

Week 14: April 26, 28

- Topics: Continuing with the object-oriented programming model taken to the extreme and Smalltalk/Squeak; a few words on scripting
- Homework 9 due, Homework 10 out

Week 15: May 3, 5

- Topics: Principles of Programming Languages; Language Generations; Closing comments; Review for Final Exam
- Homework 10 due

Final Exam:

THURSDAY, May 12, 12:40 - 2:30 pm, in BSS 308 (unless I announce otherwise)