University of Maine
School of Computing and Information Science

COS 301: Programming Languages

SYLLABUS

Fall 2017

General information

Professor: Roy M. Turner, Associate Professor of Computer Science, School of Computing and Information Science

Office hours: MW 11-12

Office: Boardman 240

Phone: 207-581-3909

Email: rturner@maine.edu

Class meetings: MWF 10:00-10:50, 203 Little Hall

Course website: MaineSAIL.umcs.maine.edu/COS301

TA: Brian Toner

Textbook: *Concepts of Programming Languages* (11^{th} edition), Robert W. Sebesta. Available in hardcopy (UMaine Bookstore, Amazon); paperback (Amazon; not completely sure this is exactly the same, so you'd have to take your chances); rental (Amazon); online via VitalSource (vitalsource.com; but it's the 10th edition at the moment). If you have the 10^{th} edition, that should be fine, as long as you're willing to look online or borrow a book from a friend for any additional material that may be missing.

Class communication: The primary means of disseminating information will be the course website and Blackboard. Announcements, links to the textbook, assignments, and so forth will be posted on the website, and grades, some on-line assignment and practice sets, etc., will be on Blackboard. I hope to set up some discussion forums (fora?) on Blackboard or the website as the semester progresses. You can contact me or the TA via email at the addresses above.

We will also use email to contact you. **The email address we will use for you is the one Mainestreet has for you.** Thus, you need to check your Mainestreet account very soon to make sure that the email address it has for you is one at which you actually receive and check email (or that it forwards to such an address).

ADA notice: If you have a disability for which you may be requesting an accommodation, please contact Disabilities Services, 121 East Annex, 581-2319, as early as possible in the term.

Emergency planning: In the event of disruption of normal classroom activities due to a disease outbreak or other long-term emergency, the format of this course may be modified to enable completion of the course (e.g., via Web-based instruction). Should such an event occur, you will be provided a revised syllabus via email and the course Web site that will supersede this version; the revised syllabus will detail the alternative format of the course.

Sexual violence/discrimination policy [University wording]: The University of Maine is committed to making campus a safe place for students. Because of this commitment, if you tell a teacher about an experience of **sexual assault, sexual harassment, stalking, relationship abuse (dating violence and domestic violence), sexual misconduct or any form of gender discrimination** involving members of the campus, **your teacher is required to report** this information to the campus Office of Sexual Assault & Violence Prevention or the Office of Equal Opportunity.

- **If you want to talk in confidence** to someone about an experience of sexual discrimination, please contact these resources:
- For confidential resources on campus: **Counseling Center: 207–581–1392** or **Cutler Health Center: 207–581–4000**.
- For confidential resources off campus: **Rape Response Services: 1–800–310–0000** or **Spruce Run: 1–800–863–9909**.
- **Other resources:** The resources listed below can offer support but may have to report the incident to others who can help:

For support services on campus: **Office of Sexual Assault & Violence Prevention: 207–581–1406**, **Office of Community Standards: 207–581–1409**, **University of Maine Police: 207–581–4040** or 911. Or see the OSAVP website for a complete list of services at `http://www.umaine.edu/osavp/`

Overview

This course covers theoretical and design principles underlying computer programming languages. A programming language is a tool for expressing problems and solutions in a formal, precise way so that a computer can carry out the solution. They are a bridge between a human programmer's way of thinking about a problem and the way a computer can carry out that problem. Computers can only "understand" machine language; a programming language provides a virtual machine to the programmer that is much closer to the way he or she thinks and that is much easier to understand and use than the raw machine.

The course will give the student a good understanding of what a programming language is, how a language's grammar is specified, and the meaning of programs written in a language. Some attention will be given to how the language is translated into machine language or interpreted to carry out the program. We will cover many aspects of imperative, object-oriented, functional, and logic programming languages.

Objectives

In general, the goals of the course are to help you achieve:

- a good understanding of what a programming language is;
- an understanding of the major language paradigms;
- a grasp of issues having to do with syntax and semantics of programs and programming languages;
- knowledge of how control and data types are handled in a variety of languages;
- knowledge of the commonalities and differences between programming languages;
- a basis for understanding how to select a programming language for a problem;
- deeper insight into programming languages you already know; and
- better professional written communication skills.

Since our BS is ABET-accredited, the accreditation goals addressed by the course are listed here:

- The student will be able to explain the major programming paradigms: imperative, object-oriented, functional and declarative.
- The student will able to recognize and explain major programming language constructs.
- The student will be able to read BNF and regular grammars, regular expressions, and graphic representations of finite state machines and will be able to write at least simple grammars, regular expressions, and FSMs.
- The student will be able to analyze a computing problem and select a language or languages that are appropriate for solving the problem
- The student will be able to recognize and describe the major historical languages and identify their influence on modern languages
- The student will learn one additional programming language
- The student will be able to apply knowledge of programming language paradigms and programming constructs to learning new languages without formal instruction
- The student will learn basic professional written communication skills and improve skills already present.
- The student will be able to write a formal research paper following a specified format with title page, table of contents, list of figures, abstract, narrative, appendices and references.
- The student will be able to use the IEEE citation standard in writing.

And the ABET outcomes addressed are:

- An ability to apply knowledge of computing and mathematics appropriate to the discipline. Students are introduced to context-free and regular grammars, finite-state automata, the fundamentals of lexical and syntactic analysis, static semantic analysis, algorithms for expression evaluation.
- An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution. Students are expected to understand how various programming languages are used in different problem domains by understanding the trade-offs between expressivity, efficiency, readability and security. Students are expected to be able to apply this knowledge to select appropriate programming languages for solving particular problems.

- An ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs. Students are expected to learn a new programming language and implement solutions to given problems in that language.
- An ability to communicate effectively with a range of audiences. This is a writing intensive course; students write five papers covering aspects of a programming language and are expected to write in a formal, professional manner with appropriate citations and an annotated bibliography. First draft papers are reviewed with comments and students then rewrite the papers, correcting issues addressed by the comments.
- Recognition of the need for and an ability to engage in continuing professional development. Students are expected to understand the historical development of programming languages from early beginnings to modern and emerging languages. Throughout the course the ability and need to learn new languages quickly is emphasized. Examples are presented in a number of languages with which students are not familiar.

Attendance

Attendance and class participation is 5% of your final grade. While I will not routinely take attendance, I reserve the right to do so. In addition, from time to time pop quizzes will be given in class both to test your understanding of current material and to record attendance.

If you cannot be present for an exam, you must notify me ahead of time and arrange to take a make-up exam, usually prior to the actual exam date. If an emergency arises and you miss class on the day of an exam, **you must contact me as soon as possible.** If you have a legitimate excuse, I may allow you to make up the exam. If you have transportation problems or oversleep, you should come to class immediately to begin taking the exam. If you do not arrive before the end of class, you should come to my office immediately.

Class participation is highly encouraged. Programming languages is a subject that can benefit from discussion, since there are many issues, opinions, and design tradeoffs involved. Student understanding, as well as class quality, can be significantly improved by active discussion in class.

Grading

Your grade will be based on:

- exams:
  - 2 preliminary exams (15% each)
  - cumulative final exam (20%)
- a semester project (35%)
- homework (10%)
- attendance and class participation (5%)

Grades will be assigned based on a common translation from percentages to letter grades: 90–100, A; 80–99, B; 70–79, C; 60–69, D; below 60, F.

There can be extenuating circumstances that cause deviations from the grading scheme described above. However, this is only done when it will raise the student's grade.

In this class, the letter grades have the usual meaning.[1] Earning an "A" means that you have done top-quality work and have excelled in meeting most course objectives. Achieving a "B" means that you have met the course objectives and have excelled in some way, for example, going beyond what is required for a "C" or exhibiting superior insight. Receiving a "C" means that you have successfully met the course objectives; a "C" is a respectable grade for an undergraduate in any course. A "D" means that you have passed, but at a low level. It should serve as a warning to you that you have not done as well as expected, that you may have trouble in computer science courses in the future, and that you are not making satisfactory progress toward your degree. An "F" means that you have not met the course objectives and have failed the course. I assign + and − grades as well to give a finer-grained evaluation of your work.

Exams

There will be two prelims and a final exam. The final will be cumulative, as will the second prelim (although most of the material covered will be since the first prelim). There will be some essay questions, and these will be graded both for content and writing. Some questions will ask you to take a position on some topic, to make arguments for or against a position, or to compare and contrast two things. Such questions generally do not have a single correct answer; you will be graded on the quality of your response, especially the arguments you provide. **No electronic devices are allowed for exams unless otherwise announced.** This includes computers, tablets, phones, calculators, and music players.

Exam questions will be individually graded on a A–F scale, which is then converted to a numeric value to use in computing the exam grade. This turns out to give students more credit than is usually the case when points are subtracted from a question's point value. On your exam, you will see a letter grade, a check mark (or "ok"), or a number beside each question. The point value earned are as follows, where n is the number of points assigned to the question:

✓ or `ok`	n	C+:	$0.78n$
A+:	$0.98n$	C:	$0.75n$
A:	$0.95n$	C-:	$0.72n$
A-:	$0.92n$	C/D:	$0.70n$
A/B:	$0.90n$	D+:	$0.68n$
B+:	$0.88n$	D:	$0.65n$
B:	$0.85n$	D-:	$0.62n$
B-:	$0.82n$	F:	$0.60n$
B/C:	$0.80n$	A number x:	x

Project

The course project is a significant part of the course. A separate document will describe the project in detail. However, briefly, the project will require involve you picking a programming language that you do not already know and writing 4–5 papers during the semester, each comparing and contrasting some aspect of or issue related to the chosen language and one you already know well.

[1] This is based on information in the *Handbook for the Faculty of Instruction University of Maine at Orono*, which was published some years ago.

You will also write programs in your chosen language. Grades will be based on both content and writing quality. It is possible that one of the paper assignments can be replaced with an in-class presentation, since this involves both written and oral communication skills. We will discuss this more in class.

Writing & editing

This is not formally a writing-intensive course, but writing is a very important part a computer scientists work, and good writing skill is a component of the outcomes expected by us and our accrediting agency for students graduating from our program. Thus, we will devote some class time to writing, and the project will be graded for writing quality (grammar, spelling, organization, etc.) as well as for content.

Part of your project will consist of "peer editing" others' projects, and you will be graded on this as well. Peer editing will help others by giving them feedback, and it will help you by forcing you to think about the elements of good writing as you look at work that is not your own, where errors are easier to see. We will discuss this more in class.

Homework

Homework assignments will be given from time to time to reinforce or provide practice thinking about topics covered in class. Should it turn out that no homework is assigned, the 10% homework component of the grade will be distributed among the exams and project.

Academic honesty

Cheating, plagiarism, or other academic misconduct will not be tolerated in this class. The minimum penalty will be a 0 on the assignment or exam in question, but egregious violations will result in referral to a University disciplinary body. You should acquaint yourself with the UMaine policy on academic integrity (umaine.edu/judicialaffairs/academic-integrity) and the UMS Student Conduct Code (linked from that page).

This policy is not meant to discourage legitimate collaboration or use of the Internet during learning. It is often helpful to study in groups and to discuss material and assignments with other students. However, you are ultimately responsible for your own work, and any work turned in should be your product. If you feel that some of the work in your assignment is actually someone else's, then you need to make this very clear in your assignment, and you would be advised to talk to me about it before you turn it in. If you have any questions about what does and does not constitute plagiarism, you need to see me before turning in the assignment. And, of course, all exam work is individual work.

Classroom behavior

You must be respectful of other students and the instructor at all times in class. Cell phones need to be turned off or silenced, and laptops or tablets should be used only for note taking or looking up material pertinent to classroom discussion. (Although I suggest that you do not use laptops for note taking, since there has been some research that shows that students retain more from taking handwritten notes.) If I feel your behavior is disruptive to class or not respectful of others, then I, *possibly* after warning you, may ask you to leave the room.

Online resources

There are many online resources related to the study of programming languages, including DMOZ.org's directory of programming languages (www.dmoz.org/Computers/Programming/Languages) and Wikipedia's comparison of programming languages (search for "comparison of programming languages). Regarding Wikipedia: articles are generally not considered authoritative and should not be used as primary sources for research papers, etc. However, Wikipedia articles can be a very good way to get an introduction to a subject and to find primary sources.

Writing resources

The Writing Center (402 Neville) is an excellent resource for help with writing.

Professor absences

Occasionally, I may have to miss class due to conferences, etc. If this should happen, then either the TA or another professor will teach that day, outside work will be assigned, a video lecture will be provided, or I may schedule make-up classes.

Schedule

The course will progress roughly in order through the book. Reading assignments will be posted on the website and announced in class. Also, see the UMaine academic calendar for important dates when there are no classes.

Acknowledgement

Thanks *very* much to Curtis Meadow, from whom I borrowed some of the material in this syllabus and much of the content of the course slides.