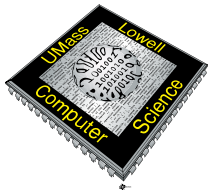

COMP3010-201 & COMP3010-202
Organization of Programming Languages
3 contact hours - 3 credits
Spring 2019 Syllabus

General Information

Instructor	Dr. Charles T. (Tom) Wilkes
Office	Dandeneau Hall 323
Email	Charles_Wilkes@uml.edu
Phone	(978) 934-3634
Class Time	(Section 201) MWF noon-12:50 PM (Section 202) MWF 1:00-1:50 PM
Class Location	(Section 201) Kitson Hall 303 (Section 202) Ball Hall 210
Office Hours	MWF 2:00-5:00 PM (with exceptions for CS Department faculty meetings, etc., as announced)
Teaching Assistant	TBD

Required Textbook

- Michael L. Scott, *Programming Language Pragmatics* (4th edition), Wiley

Supplementary Textbook / Materials

- Required textbook's Companion Site

Course Description

- This course presents an analytical approach to the study of programming languages. It provides a description of the salient features of the imperative, functional, logical, and object-oriented programming paradigms in a suitable metalanguage such as Scheme. Topics include iteration, recursion, higher-order functions, types, inheritance, unification, message passing, orders of evaluation, and scope rules; elementary syntactic and semantic descriptions; and implementation of simple interpreters.

Course Prerequisites

- COMP2010 Computing III

Course Category

- Required

Additional Course Information

- **Topics** – will include:
  - Formal aspects of syntax and semantics
  - Naming, scoping, and binding
  - Scanning, parsing, semantic analysis, and code generation
  - Control flow, subroutines, exception handling, and concurrency
  - Type systems, composite types, data abstraction, and storage management
  - Imperative, functional, logic-based, and object-oriented programming paradigms
- **Purpose** – As the author of our required textbook expresses it:
  "[This course] is an introduction to the design and implementation of programming languages. From the design point of view, we will study language features as tools for expressing algorithms. From the implementation point of view, we will study compilers, interpreters, and virtual machines as tools to map those features efficiently onto modern computer hardware. The course will touch on a wide variety of languages, both past and present, with an emphasis on modern imperative languages, such as C++ and Java, and, to a lesser extent, on functional languages such as Scheme and ML; scripting languages such as Perl, Python, and Ruby; and emerging languages such as Scala, Rust, and Swift. Rather than dwell on the features of any particular language, we will focus instead on fundamental concepts, and on the differences between languages, the reasons for those differences, and the implications those differences have for language implementation."
- **Web Page** – on Blackboard
- **Software** – Linux virtual machine (provided by the instructor) hosted on Oracle VM VirtualBox

Methodology

Teaching methods: This course will make use of lectures, examples, and question and answer sessions. Hands-on programming activities using computers during class, in addition to active reading, experimenting with sample programs, and problem solving, will be used during the semester. Independent programming assignments are used to evaluate language and analytical skills.

Assessment: There will be in-class exercises, regular programming practice problems and quizzes, programming exercises, two in-class written exams, and a comprehensive final exam.

Grading:

The percentage weights of the various assignments in this class are as follows:

Program Average	40%
Quiz Average	10%
In-Class Exams	2x15%
Final Exam	20%

At the end of the semester, a course average is computed for each student using the percentage weights, and the final grade is assigned using the following scale:

A 90.0-100

B+ 87.0-89.9
B 83.0-86.9
B- 80.0-82.9

C+ 77.0-79.9
C 73.0-76.9
C- 70.0-72.9

D+ 67.0-69.9
D 60.0-66.9

F below 60

Factors such as attendance, improvement (especially as demonstrated on the final exam), and class participation may be considered when making the final decision in borderline cases.

Requests for re-grading assignments may be made up to one week after the assignment is returned. The request must be submitted in writing and include a short paragraph outlining the rationale for the re-grade. Acceptable requests include correcting errors in calculating a score, marking a correct answer incorrect, etc. (See also "Resubmission Policy" below.)

University Policies and Advising Resources

Students are expected to be familiar with the following policies from the Undergraduate Catalog:

- Academic Policies: https://www.uml.edu/Catalog/Undergraduate/Policies/Academic-Policies/Academic-Policies.aspx
- Kennedy College of Sciences Policies: https://www.uml.edu/Catalog/Undergraduate/Sciences/policy/default.aspx
- Computer Science Department Policies: https://www.uml.edu/Catalog/Undergraduate/Sciences/policy/Continuance-appeal-dismissal.aspx

Also, students are urged to be aware of advising resources provided by the University:

- https://www.uml.edu/Academics/Provost-office/Faculty-success/Professional-Engagement/advising-resources.aspx

Course Policies

Attendance: Formal attendance is mandatory. Three unexcused absences are permitted. Additional absences may result in a failing grade and/or removal from the course.

Academic integrity: The practice of good ethical behavior is essential for maintaining good order in the classroom, providing an enriching learning experience for students, and as training as a practicing computing professional upon graduation. This practice is manifested in the University's Academic Integrity policy. Students are expected to strictly avoid academic dishonesty and adhere to the Academic Integrity policy as outlined in the course catalog. Violations will be dealt with as outlined therein.

As a general rule, all work submitted for grading must be the student's own work. Students are allowed to help each other solve compiling and linking problems, and may generally discuss issues related to a student's particular program, but students may not share code, write code, or examine another student's code.

In regard to homework and projects, students may discuss the problems (what is being asked for), appropriate material from class lectures or the textbook or acceptable other sources. Students, however, may not share answers or the specifics of how to answer the question.

Use of material from previous classes, solution manuals, material from the Internet or other sources (e.g., parents, siblings, friends, etc.) that directly bears on the answer is strictly prohibited.

At the discretion of the instructor, students may be asked to sign a statement that they have abided by the University's Academic Integrity policy and its application to this class. This statement may appear on homework, tests, or projects.

When in doubt, consult the course professor before doing something that may result in violation of the University's Academic Integrity policy.

Application to this course:
Programming assignments are to be done by the student alone. No outside help is permitted. If you need help on a programming assignment you can only receive aid from the instructor of the course, teaching assistant or approved tutors.

The sanction for the first violation of the Academic Integrity policy or plagiarism policy will result in a minimum failing grade on the relevant assignment and the violation will be reported to the student's department chair. Once the final decision has been rendered and any or all appeals exhausted by any parties involved, the instructor or appropriate parties will carry out the recommended sanction.

Personal conduct: In order to minimize distractions and interruptions, students will be expected to:

1. Arrive to class on-time and fully prepared.

2. Give the instructor full and undivided attention once the lecture has begun.

3. Turn off and stow all cell phones, pagers, and any other personal electronic devices once the lecture has begun.

Failure to adhere to these policies may result in immediate dismissal from class and loss of any in-class credit for relevant assignments or activities.

Computer use: Students are encouraged to use their laptop computer for taking notes and other activities directly related to the course. The manner in which students use the computer in class is considered a matter of honor and professionalism. Students will adhere to the following guidelines:

1. Computer use must be for taking notes or other activities related directly to the course.

2. Computer use must be subtle and must not distract fellow classmates or the instructor.

Inappropriate use of a computer in the classroom will be viewed as disrespectful to the instructor and classmates and will be considered unprofessional. Examples of inappropriate use include, but are not limited to:

- Sending, receiving, or reading e-mail
- Instant messaging
- Web browsing
- Working on assignments
- Playing games
- Listening to music
- Watching movies

Judgments regarding the appropriateness of computer use are at the discretion of the instructor. The consequences for violating this policy are also at the discretion of the instructor, and may include loss of in-class computer privileges, grade reduction, and so forth.

Course readings: Readings are to be completed before the class session for which they have been assigned; material covered in each reading is fair game for class discussions and unannounced quizzes.

Assignments: Assignments will be distributed via a mixture of paper and electronic means. Students are responsible for managing due dates and understanding submission procedures to turn in programming assignments.
Exceptions to assignment due dates will be made only in very dire circumstances (see "Late work" below); therefore, please submit whatever you have by the due date—even if it's only reasonable preliminary thoughts or pseudocode!—so that the instructor can give you at least some partial credit instead of a zero on the assignment.

Resubmission Policy: Assignments may be submitted an unlimited number of times before the due date; only the last submission will be graded. An assignment may be resubmitted up to one week after the student receives comments from the instructor or grader concerning the assignment.

Exams: A discussion of the topics covered by each exam will occur during class at most one week from the scheduled exam date.

Final exam: The comprehensive final exam for this class currently is scheduled as follows:
Section 201 – Friday, May 10, 8:00-11:00 AM
Section 202 – Monday, May 6, 3:00-6:00 PM
Please double-check your final exam schedule on SiS.

Late work: Late assignments or projects will not be accepted without prior approval. Students must consult the instructor at least three days prior to the scheduled due date of any assignment to make alternative arrangements; however, the instructor is under no obligation to grant any such request. Penalties such as a reduced score may be applied at the instructor's discretion.

In-class exams will not be rescheduled except in the event of an excused absence. The instructor will work with the student to schedule such a makeup exam; however, it is in the student's best interest not to miss the regularly scheduled in-class exam.

Tentative Schedule

Week	Topic	Readings*	Assignments
1 (1/23-1/25)	Introduction Structure of a compiler	PLP 1.0–1.5 PLP 1.6–1.7	A0: Unix tools (no submission) A1: Comparing languages
2 (1/28-2/1)	**Syntax**: Review of scanning and BNF (DFA construction) **Syntax:** Top-down and bottom-up parsing	PLP 2.0–2.2; CS 2.4.0–2.4.1 PLP 2.3.0–2.3.2; CS 2.3.5 (pp. 1–6)	
3 (2/4-2/8)	**Syntax**: Error recovery, table-driven LL(1) parsing **Semantics**: Attribute grammars, action routines	PLP 2.3.3 PLP 4.0–4.5	A1 due A2: Syntax error recovery (see CS 2.3.5 pp. 10–12)
4 (2/11-2/15)	**Semantics:** Decorating a syntax tree **Functional programming:** Intro to OCaml	PLP 4.6-4.7, 15.0-15.3 PLP 11.0-11.2, 7.2.4, 8.6, 11.4	
5 (2/19-2/22)	**Functional programming:** Evaluation order, higher-order functions **Compiler internals:** Intermediate representations	PLP 11.5-11.6, 11.8-11.9 PLP 15.1-15.2; CS 15.2.1	**Monday, February 18: President's Day Holiday** **Tuesday, February 19: Monday schedule** A2 due A3: Translation (in OCaml)
6 (2/25-3/1)	**Scripting languages:** Overview, Perl **Friday, March 1: Exam 1**	PLP 14.0-14.1, 14.2.4 (Perl), 14.4.2, 14.4.3	
7 (3/4-3/8)	**Names:** Binding time, scope rules **Names:** Binding rules, separate compilation	PLP 3.0-3.4 PLP 3.5-3.9; CS 3.8	A3 due A4: Cross-indexer **3/11-3/15: Spring Break**

8 (3/18-3/22)	**Control flow:** Expressions, sequencing, and selection **Control flow:** Iteration and recursion	PLP 6.0-6.4 PLP 6.5-6.8	
9 (3/25-3/29)	**Type systems and composite types:** Type checking **Type systems and composite types:** Records and arrays	PLP 7.0-7.2 PLP 8.0-8.4	
10 (4/1-4/5)	**Type systems and composite types:** Pointers, references, and dynamic storage management **Type systems and composite types:** Polymorphism	PLP 8.5 PLP 7.3-7.5; CS 7.3.2	A4 due A5: Tombstones
11 (4/8-4/12)	**Concurrency:** Threads and synchronization **Concurrency:** Language mechanisms	PLP 13.0-13.2.3, 9.5, 13.2.4-13.3 PLP 13.4	
12 (4/17-4/19)	(Catch-up: No new reading) **Friday, April 20: Review for Exam 2**		**Monday, April 15: Patriot's Day Holiday** A5 due A6: Concurrency (in Java)
13 (4/22-4/26)	**Monday, April 22: Exam 2** **Subroutines:** Stack management, parameter passing **Subroutines:** Exceptions, events	PLP 9.0-9.3 PLP 9.4.3, 9.6	
14 (4/29-5/3)	**Objects:** Review and implementation **Objects:** Multiple inheritance; Smalltalk, scripting objects **Reading Day: Saturday, May 4** **Final Exam:** **Section 201 – Friday, May 10, 8:00-11:00 AM** **Section 202 – Monday, May 6, 3:00-6:00 PM**	PLP Chapter 10 (skim 10.1-10.3, but concentrate on 10.4) CS 10.5, 10.6.1; PLP 14.4.4	A6 due

*** "PLP" refers to the required textbook; "CS" refers to the textbook's Companion Site.**