University of South Carolina
Department of Computer Science and Engineering
College of Engineering and Computing

Programming Language Structures
CSCE 330 Section 001 Fall 2023

Instructor	Dr. Marco Valtorta
Course Websites	https://cse.sc.edu/~mgv/csce330f23/index.html (main), https://blackboard.sc.edu (lectures if recorded), https://dropbox.cse.sc.edu (assignments)
Phone	(office) 803-777-4641
Email	mgv@cse.sc.edu
Office location	Innovation Center (INNOVA) 2269
Teaching Assistant	Qiaoyu Liang, qliang@email.sc.edu
Meeting Times and Place	TTh 1:15-2:30pm SWGN 2A14 Office Hours: Tuesday and Thursday 1445-1615 in my office. Please email mgv@cse.sc.edu for individual consultation; if necessary, an in-person, phone or virtual face-to-face conversation will be arranged. Office hours for Mr. Liang: TTh 0900-1000 in INNOVA 1211

NOTE: I check email very frequently and usually respond within a few hours. Email is the preferred and strongly recommended means of communication with the instructor.

Academic Bulletin Description
CSCE 330 -- Programming Language Structures (3 Credits)
Formal specification of syntax and semantics; structure of algorithms; list processing and string manipulation languages; statement types, control structures, and interfacing procedures.
Prerequisites: CSCE 240; MATH 174 or MATH 374 or MATH 574.

Course Description: History and families of programming languages. Programming in Prolog at an introductory level: relations, recursion, backchaining, constraint processing, lists. Programming in Haskell at an introductory level: recursive and higher-order functions, list comprehensions, types and classes. Syntax and semantics of programming languages.

Pre-requisites: CSCE 240; MATH 174 or MATH 374 or MATH 574.

Carolina Core Learning Outcome: None.

Credits for this course: 3

Course Learning Outcomes
Upon successful completion of this course, students should be able to:

- Categorize a language as imperative (procedural), functional (applicative) or declarative (logic)
- Generate and use syntax descriptions in EBNF
- Write code in a logic language (e.g., Prolog)
- Write code in a functional programming language (e.g., Haskell)

Course Overview
This course will be delivered in a lecture room.

• Student-to-Instructor (S2I) Interaction: Students will listen/view lectures in class. The professor will hold office hours, post announcements on the course websites, and provide individual feedback to students through the CSE dropbox website and email.

• Students-to-Student (S2S) Interaction: Students will engage in discussions through email and the discussion forum on Blackboard.

• Student-to-Content (S2C) Interaction: Students will engage with course content by completing assignments and reports and participating in lecture meetings.

The instructor will reply to all feedback in a reasonable amount of time; the same is expected of the students. Specifically,

• Communication: Responses to email communication and questions will normally be provided within 48 hours.

• Assignment and Test Grading Grades for assignments will be returned within one week of due date.

• All assignments are due before the beginning of class. A 10% deduction will be given to late submissions that are turned in before the beginning of the next class. No credit will be given for assignments that are turned in after the beginning of the next class.

Required Textbook:
- Hector J. Levesque. *Thinking as Computation*. The MIT Press, 2012, ISBN 978-0-262-01699-5. This text is referred to as [L]. Supplementary materials from the author are available.
- Graham Hutton. *Programming in Haskell, 2nd Ed*. Cambridge University Press, 2016, ISBN: 978-1-316-62622-1. This text is referred to as [H] in the syllabus (including course assignments, lecture log, etc.). The author maintains a website for this textbook, which includes an errata list, at http://www.cs.nott.ac.uk/~pszgmh/pih.html

Additional Materials That May be Used:
- Chapter 2 of: Ghezzi, Carlo, and Mehdi Jazayeri. *Programming Language Concepts, 3rd ed.* Wiley, 1998 (referred to as [G] or [G&J]).
- Section 7.3, Chapter 8, and Chapter 9 of: Ghezzi, Carlo and Mehdi Jazayeri. *Programming Language Concepts, 2nd ed.* Wiley, 1987 (referred to [G&J, 1987]).

- Chapter 2 of: Allen Tucker and Robert Noonan. *Programming Languages: Principles and Paradigms, 2nd ed.* McGraw-Hill, 2007 (referred to as [T]).
- John Backus. "Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs (1977 Turing Award Lecture)." Communications of the ACM, vol. 21, issue 8, pp.613-641, August 1978 (referred to as [B]).
- The main course website has links to other papers, reports, etc., that may be used.

Technology and Required Course Materials:

PowerPoint Presentations + Textbook + Computer with Internet Access + Access to a Computer with SWI-Prolog and the Haskell GHC compiler installed.

- Students must view PowerPoint lectures/presentations.
- Students must read chapters in the required textbook.
- Students must have access to a computer with Internet access to check the course website maintained by the instructor (https://cse.sc.edu/~mgv/csce330f23), the departmental dropbox (https://dropbox.cse.sc.edu/login), and Blackboard.
- Students must that the ability to create, save and upload programs to the departmental dropbox.
- Students must have access to a computer with SWI-Prolog installed.
- Students must have access to a computer with the GHC Haskell compiler installed.

Course Purpose and Overall Structure of the Course:

The course has four parts. The first part is a historical and conceptual overview of programming languages, including a brief introduction to programming language families. The second part is a primer on Prolog, the most widely used logic language. Logic languages were introduced in the 1970s and are very high-level declarative languages, related to knowledge representation languages in Artificial Intelligence. We will emphasize a relational view of logic languages. The third part is a primer for a modern functional programming language, Haskell. Functional programming has a long history, based on Alonzo Church's work on the lambda calculus in the 1930s and John McCarthy's LISP interpreter of the late 1950s. It has seen a renaissance with the emergence of the ML family of languages, culminating with the Haskell programming language. Functional programming is now used in industry as well as in academia, with advantages such as: strong static typing and type inference, the ability to treat functions as first-order objects in combining forms, persistent data structure, equational theories that allow for proof of correctness via program transformation and incremental development of efficient programs, etc. Many mainstream programming languages now include some functional features, such as anonymous functions and higher-order functions. Still, the ability to think functionally and express oneself idiomatically requires the discipline of using a (nearly) pure functional language, such as Haskell. The fourth part of the course is a discussion of syntax, emphasizing regular and context-free languages, and semantics, emphasizing the basics of axiomatic (Hoare-style), denotational, and operational semantics. This presentation will not include formal definitions and proofs.

Technical Support:

Blackboard Help (http://ondemand.blackboard.com/students.htm)

If you have problems with your computer or Blackboard, please contact University Technology Support (UTS) Help Desk at 803.777.1800 or helpdesk@sc.edu. The UTS Help Desk is open Monday – Friday from 8:00 AM – 6:00 PM. If you have problems with the departmental dropbox, please contact the instructor. The departmental and college computers are always available online and include the required Haskell GHC compiler.

Late Work/Make-Up Policy:
Late work is accepted with a 10% penalty until the beginning of the class after the due date.

> **BE CAREFUL:** The clock on your computer may be different than that clock on Blackboard. If the clock is different by one minute, you might be subject to the late homework penalty. Plan accordingly. I recommend that you submit your assignments well before the deadline.

Extra Credit:

Extra credit assignments will not be assigned.

Attendance Policy:

The attendance policy in the University Academic Bulletin ("Attendance Policy" section of https://academicbulletins.sc.edu/undergraduate/policies-regulations/undergraduate-academic-regulations/#text) is followed in this course, and students must request excuses in the way described there and abide by the "5% Rule" described there. Failure to abide by the policy will not result in a grade penalty, but please see the description of quizzes in the section "Grades Will Be Calculated as Follows." In addition, students are very strongly encouraged to attend every class. I estimate one missed class to result in at least three hours of extra work outside of class.

Ability to Work at Your Own Pace:

The course builds upon the material in an incremental fashion. It is difficult to catch up if several classes are missed.

The course syllabus includes an accessibility statement that encourages students with disabilities to register with the Student Disability Resource Center; should a student with a registered disability enroll in the course, the professor will work with the Office to make any additional accommodations appropriate to that student's needs.

Course Communications:

You are required to use your *UofSC email account* throughout this course. I will be communicating with you regarding grades and assignments. I will reply to emails within 24 hours and will provide feedback on assignments within 48 hours. When sending an email, please include a *detailed subject line*. Additionally, make sure you reference the course (CSCE 330) and section (001) and sign the email with your name. Begin these emails with a proper salutation (e.g., Dear Dr. Valtorta, Hello Dr. Valtorta, and Good evening Dr. Valtorta). Starting an email without a salutation or a simple "Hey" is not professional or appropriate. Of course, if the emails evolve into a thread with rapid exchanges, e.g. when discussion a programming issue, you may omit the salutation.

Grades Will Be Calculated as Follows

Most of the assignments are programming exercises. Overall, the exercises will count for 45% of the grade. A midterm exam will count for 15% of the grade, and a final exam will count for 30% of the grade. There will be a presentation worth 5% of the grade and quizzes also worth 5% of the grade. The conversion from percentage to letter is the standard one, with 10-point intervals for each letter and seven points required for a plus; the overall numeric score will be rounded up to the nearest integer. A combined score of 60% on the midterm and the final is required to obtain a C in the course; a student who would otherwise obtain a C or better in the course will obtain at most a D+ if his or her combined score on the midterm and final is less than 60%; this percentage is computed as a weighted average of the percentages in midterm and final, with the final weighing twice the midterm. Simple grading rubrics will be posted on the instructor's course website under "points per assignment." Please review the rubric before starting the assignments.

Quizzes are treated in an unusual way in this course. They are often in class exercises designed to emphasize a salient issue in the lecture of the day or in a previous lecture. Another purpose of quizzes is to take attendance. Students who take all quizzes and who, in the judgement of the instructor, make an honest attempt to answer the quiz will receive full credit for them. Missed quizzes may be replaced by correctly answered quizzes, up to a number (usually three), which is determined by the instructor, and depends on how many quizzes are given.

Warning – on any written assignment and, especially, programming assignments: Please do not plagiarize. I normally do not use an automated system to detect plagiarism, but this may change at any time.

Disability and Other Student Support Services:

Students with disabilities should contact the Student Disability Resource Center. The contact information is below:

	Phone: 803-777-6142
Close-Hipp Suite 102	Fax: 803-777-6741
1705 College Street	Email: sadrc@mailbox.sc.edu
Columbia, SC 29208	Web: Student Disability Resource Center

These services can aid with accessibility and other issues to help those with disabilities be more successful in the course. Additionally, students with disabilities should review the information on the Disabilities Services website and proactively communicate with the professor before or during the first week of class.

The following other academic support services and resources may help you be more successful in the course as well.

Library Services (http://www.sc.edu/study/libraries_and_collections)
Writing Center (http://artsandsciences.sc.edu/write/)
Student Technology Resources
(http://www.sc.edu/about/offices_and_divisions/division_of_information_technology/)

Academic Honesty:

Every student has a role in maintaining the academic reputation of the university. It is imperative that you refrain from engaging in plagiarism, cheating, falsifying your work and/or assisting other students in violating the Honor Code.

Plagiarism/Cheating, as defined in the Code of Student Academic Responsibility, will **result in failure** of this course in addition to any penalty/penalties exacted by the appropriate Academic Dean and the University Honor Council to whom all offenses will be reported. Consult *Office of Academic Integrity* for what constitutes plagiarism. You are responsible for reading and abiding by these rules. The websites provided below should be reviewed so you can learn more about the University policies.

- Student Conduct and Academic Integrity (https://sc.edu/about/offices_and_divisions/student_conduct_and_academic_integrity/index.php)
- Carolinian Creed (https://sc.edu/about/offices_and_divisions/student_affairs/our_initiatives/involvement_and_leadership/carolinian_creed/index.php)
- The Honor Code (http://www.sc.edu/policies/staf625.pdf)
- Guidelines for Responsible Computing: (https://www.sc.edu/about/offices_and_divisions/university_technology_services/policies_procedures/networkguideline.php)

You must save files on a USB drive or other secure area. Many of you are familiar with github from other courses or have used other version control or code management systems (e.g.

bitbucket). The programming assignments in this course are rather short and do not build on each other, so you are not required to use such a system but doing so may be useful for creating an electronic portfolio as you start your careers. Do not save your work on a public computer or library computer as others will have access to your work. You should have a back-up of all your files in another location. Submitting someone else's work is cheating and against the Carolina Code. Cheating will result in penalties**.** All parties will also be referred to the Office of Academic Integrity for additional retribution. One or more of the sanctions described at https://sc.edu/about/offices_and_divisions/student_conduct_and_academic_integrity/hearings/case_outcomes/honor_code_sanctions/index.php may be imposed.

It is very good to study in groups. In fact, there is evidence that group studying is a predictor of success, at least in early college mathematics courses. Some of you may enjoy studying in groups! You are therefore encouraged to discuss the material you study, but you must do your homework individually, unless an assignment is explicitly designated as a team assignment. The minimum grade penalty for a violation will be a zero on the work involved. In addition, an honor code violation will be subject to the sanctions described in the USC Community Handbook and Policy Guide. The following paragraph, written by Professor Duncan Buell, clarifies the distinction between "learning from a discussion" and "turning in someone else's work": If, after having participated in a group activity, you can walk away, put the books down, have lunch, and then come back afterwards to re-create from your own head the material and techniques you discussed as a group, then you can legitimately say that you have learned from the group but the work you turn in is your own.

CSCE 330 --- FALL 2023
TIME ALLOCATION FRAMEWORK

WEEK	TOPIC	SOURCE
1 (lecture 1; 8/24, Thursday Only)	Introduction and Historical Review	Notes
2 (2,3; 8/29,31)	Historical Review; The Prolog Language	Chs.1-2 [L]
3 (4,5; 9/5.7)	The Prolog Language	Chs.2-3 [L]
4 (6,7; 9/12,14)	Writing Prolog Programs	Ch..4 [L]
5 (8,9; 9/19,21)	Case Study: Satisfying Constraints	Ch.5 [L]
6 (10,11; 9/26,28)	Case Studies: Satisfying Constraints and Interpreting Visual Scenes	Chs.5, 6 [L]
7 (12,13; 10/3,5)	Lists in Prolog	Ch.7 [L]
8 (14,15;10/10,12)	Review and Midterm	
9 (16; 10/17, Tuesday Only)	Programming Language Syntax (Thursday: Fall Break)	Notes based on [T]
10 (17,18; 10/24,26)	Haskell: Introduction, Types and Classes	Chs.1-3 [H]
11(19,20; 10/31,11/2)	Defining Functions; List Comprehensions	Chs.4-5[H]
12 (21,22; 11/7,9)	Recursive and Higher-Order Functions	Chs.6 and 7 [H]
13 (23,24; 11/14,16)	Backus's FP; Declaring Types and Classes	[B] and Ch.8 [H]
14 (25; 11/21, Tuesday Only)	The Countdown Problem	Ch. 9 [H]
15 (26,27; 11/28,30)	Programming Language Semantics; Student Presentations	[G&J; G&J,1987]
16 (29,30; 12/5,7)	Student Presentations	
Tuesday, 12/12, 12:30	Final Exam, in the classroom	