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

Topics in Information Technology: Functional Programming
CSCE 590 Section 001 Location: Online – Fall 2020

Instructor	Dr. Marco Valtorta
Course Websites	https://cse.sc.edu/~mgv/csce590f20/index.html (main), https://blackboard.sc.edu (lectures), https://dropbox.cse.sc.edu (assignments)
Phone	(office) 803-777-4641; (mobile) 803-446-3225
Email	mgv@cse.sc.edu
Office location	Innovation Center (INNOVA) 2269
Meeting Times	TTh 1005-1120. (on Blackboard Collaborate Ultra)
	Office Hours: Monday 1500-1800 on Blackboard Collaborate Ultra. Please email mgv@cse.sc.edu for individual consultation; if necessary, a phone or virtual face-to-face conversation will be arranged.

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 590 - Topics in Information Technology: Reading and research on selected topics in information technology. Course content varies and will be announced in the schedule of courses by suffix and title. May be repeated for credit as topics vary. Credits: 3.

Course Description: Functional programming as a paradigm. History of functional programming languages: Lisp, FP/FL, Scheme, ML, Miranda, Haskell. Programming in Haskell at an intermediate level: recursive and higher-order functions, list comprehensions, types and classes, monads, lazy evaluation, reasoning about programs.

Pre-requisites: CSCE 330 or CSCE 350 or MATH 374.

Carolina Core Learning Outcome: None.

Credits for this course: 3

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

- Demonstrate the ability to design a functional programming solution to algorithmic problems
- Demonstrate the ability to write programs of intermediate complexity in Haskell
- Demonstrate the ability to prove the correctness of simple functional programs

Graduate students will also be able to demonstrate the ability to evaluate an important paper from the peer-reviewed literature on functional programming and assess its contribution by writing a report and preparing a presentation.

All learning outcomes in this Distributed Learning course are equivalent to those in the face-to-face (F2F) version of this course.

Course Overview
This course will be delivered synchronously through Blackboard.

- Student-to-Instructor (S2I) Interaction: Students will listen/view lectures online and interact with the professor via Blackboard Collaborate Ultra. The professor will hold online office hours via Blackboard Collaborate Ultra, 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 video conference 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 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:

- 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

Recommended Textbooks:

[TFWH] Richard Bird. *Thinking Functionally with Haskell*. Cambridge University Press, 2015, ISBN: 978-1-107-08720-0 (Hardback), 978-1-107-45264-0 (Paperback). This book is very strongly recommended; an electronic version can be rented from amazon.com.

[RWH] Bryan O'Sullivan, John Goertzen, and Don Stewart. Real World Haskell. O'Reilly Media, 2009, ISBN: 978-0-596-51498-3. The text is available for free (with comments) at http://book.realworldhaskell.org/

Technology and Required Course Materials:

Video PowerPoint Presentations + Textbook + Computer with Internet Access + Access to a Computer with the Haskell Platform 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/csce590f20), 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.
- Studenst must have access to a computer with the Haskell Platform (including the GHC compiler) installed.

Course Purpose and Overall Structure of the Course:

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, where advantages such as: strong static typing and type inference, the ability to treat functions as first-order objects in combining forms, persistent data structure, strong 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.

Even though functional programming is not emphasized in our Computer Science curriculum, this is a traditional programming course, with lectures and programming assignments. It is not a software engineering course and it is concerned more with the essence of algorithm development than with important ancillary issues such as file processing, so the programs assigned are rather short. Still, Haskell is a very concise and high-level language, so a lot can be done with few lines of code.

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:

There is no penalty for missing classes. 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 Office of Student Disability services; 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 590) 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

The plan is to complete most of the exercises in [H]. Most of the exercises are programming exercises. Overall, the exercises will count for 55% of the grade. A midterm exam will count for 15% of the grade, and a final exam will count for 30% 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. 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 may 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.

Graduate Student Assessment

Graduate students will also be required to write a 10-page report and present a 15-minute presentation with pdf or PowerPoint slides, describing and evaluating a paper from the peer-reviewed literature. A particularly good source for such presentations is the collection of "Functional Pearls" articles in the *Journal of Functional Programming*. Also see https://wiki.haskell.org/Research_papers/Functional_pearls and: Richard Bird. Pearls of Functional Algorithm Design. Cambridge University Press, 2010, ISBN 978-0521513388. This additional work is not part of the numeric grade calculation, but it is required and worth up to a letter grade. A graduate student who does not complete the report and the presentation will lose a full letter grade. A more detailed rubric for evaluation of the report and presentation will be provided later.

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:

1523 Greene Street
LeConte Room 112A
Columbia, SC 29208

Phone: 803-777-6142
Fax: 803-777-6741
Email: sadrc@mailbox.sc.edu
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.

- Carolina Community (http://www.sa.sc.edu/carolinacommunity/)
- Carolina Creed (http://www.sa.sc.edu/creed)
- Academic Responsibility (http://www.sc.edu/policies/staf625.pdf)
- Honor Code Violations: (https://www.sa.sc.edu/academicintegrity/sanctions-2/)
- 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 following sanctions may be imposed. Office of Academic Integrity (https://www.sa.sc.edu/academicintegrity/sanctions-2/)

- Expulsion from the University.
- Suspension from the University for a period of no less than one semester.
- Probation. A period of review and observation during which a student is under an official notice that subsequent violations of the Honor Code are likely to result in a more sever sanction including suspension or expulsion from the university.
- Written Warning (first offense only). An official reprimand that makes the misconduct a matter of record in University files. Any further misconduct could result in further disciplinary action.
- "X" on the transcript before a grade denoting an Honor Code Violation.
- Academic Integrity Workshop.
- Research Project. This sanction typically should be assigned for the educational benefit of the students and should be related to academic integrity or ethics overall or in the discipline in which the offense occurred. They will be monitored by the Office of Academic Integrity.
- A combination of the above sanctions.

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 590 --- FALL 2020
TIME ALLOCATION FRAMEWORK

WEEK	TOPIC	SOURCE
1	Introduction and the GHC Compiler and Haskell Platform	Chs. 1 and 2 [H]
2	Types and Classes	Ch.3 [H]
3	Defining Functions and List Comprehensions	Chs. 4 and 5 [H]
4	Recursive Functions	Ch. 6 [H]
5	Higher-Order Functions	[B] and Ch.7 [H]
6	Declaring Types and Classes and the Countdown Problem	Chs. 8 and 9 [H]
7	Review and Midterm	
8	Interactive Programming	Ch. 10 [H]
9	Two-person Games	Ch. 11 [H]
10	Functors, Applicatives, and Monads	Ch. 12 [H]
11	Monadic Parsing	Ch. 13 [H]
12	Foldables and Lazy Evaluation	Chs. 14 and 15 [H]
13	Reasoning about Programs	Chs. 16 and 17 [H]
14	Functional (Persistent) Data Structures	Instructor's Notes
	Final Exam: December 10, 9 a.m., in the classroom	

Additional Reference:
[B]. 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.
NOTE: Comments on the history of functional programming and discussions on comparative language issues will happen throughout the course.