Syllabus

General Information

Instructor: Matthew Fluet  
E-mail: mtf@cs.rit.edu  
Office hours: MW 9:00am – 11:00am; GOL-3555  
or by appointment

Lectures:  
Section 01  
MW 2:00pm – 3:50pm; GOL-3455

Website:  
http://www.cs.rit.edu/~mtf/teaching/20123/psfp  
http://mycourses.rit.edu

Course Description

The goal of this course is to introduce the students to a programming paradigm and an appropriate programming language chosen from those that are currently important in industry or that show high promise of becoming important. A significant portion of the learning curve occurs through programming assignments with exemplary solutions discussed later in class. [4005-714: Students must complete a separate term project which will require some skills not discussed in class.] The instructor will post specifics prior to registration. With the approval of the program coordinator, this course can be taken for credit more than once, provided each instance deals with a different paradigm and language.

This course instance is “Functional Programming and Haskell”.

Prerequisites

- (official) 4003-561: 4003-233 (Computer Science 3) or 4003-243 (Object-Oriented Programming in Java) or 4003-236 (Computer Science for AP Students) or 4003-263 (Computer Science for Transfers)
- (official) 4005-714: 4003-707 (Advanced Programming)
- (recommended) 4003-561: 4003-450 (Programming Language Concepts)
- (recommended) 4005-714: 4003-709 (Programming Language Concepts)
- or permission of instructor
Text Books

Required:
Title: Haskell: The Craft of Functional Programming (3rd edition)
Author: Simon Thompson
Publisher: Addison-Wesley/Pearson
ISBN: 978-0201882957
Website: http://www.haskellcraft.com/craft3e/Home.html

Suggested:
Title: Programming in Haskell (7th edition)
Author: Graham Hutton
Publisher: Cambridge University Press
ISBN: 978-0521692694
Website: http://www.cs.nott.ac.uk/~gmh/book.html

Grades, Exams, and Assignments

Grades will be assigned based on the following grading scheme:

<table>
<thead>
<tr>
<th></th>
<th>4003-561</th>
<th>4005-714</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attendance &amp; Participation:</td>
<td>5%</td>
<td>4%</td>
</tr>
<tr>
<td>Homework Assignments (≈ 8):</td>
<td>65%</td>
<td>55%</td>
</tr>
<tr>
<td>Project:</td>
<td>13%</td>
<td></td>
</tr>
<tr>
<td>Mid-term Exam:</td>
<td>15%</td>
<td>14%</td>
</tr>
<tr>
<td>Final Exam:</td>
<td>15%</td>
<td>14%</td>
</tr>
</tbody>
</table>

Attendance & Participation

Students are required to attend and expected to participate in class. Participation means being an engaged student: asking and answering questions, not simply attending class.

The use of cell phones and audio players is prohibited during class. If you must take a phone call, please leave the classroom immediately and do not return until you have ended the phone call.

The use of a laptop (or notebook or netbook) computer is permitted during class only for the purpose of taking notes. Persistent use of a laptop for other activities will result in 0 credit for your Attendance & Participation grade.

Assigned readings should be completed before the lecture section. You are responsible for the material in assigned readings, whether covered during lecture or not.

Project (4005-714 only)

Students taking the course for graduate credit will complete a “project” in order to explore an aspect of functional programming not (extensively) discussed in lecture. The project consists of choosing a topic, researching and exploring the topic, writing a report, and developing a software artifact. The project may be seen as creating a
“course module”: your topic should be of a size and scope suitable for presentation in one or two lectures (though, you will not need to give a presentation), your report should serve as lecture notes on the topic, and your software artifact should serve as an extended example and/or assignment on the topic.

Suggested Topics

- explore another functional programming language
- explore an advanced feature of Haskell/GHC
- explore a significant Haskell or SML library
- explore a conference paper or journal article

These are broad topic categories; you will be expected to select a specific topic, which must be submitted to and approved by the instructor. Researching your chosen topic should include reading sufficient background and primary source material on the topic (book, article, paper, or manual; Wikipedia is not a primary source material) to confidently understand and explain the topic.

Report and Software

In the context of your chosen topic, you will write a report and develop a software artifact.

The report should include: a thorough description of your chosen topic (with sufficient background and examples); a thorough description of your software artifact and how it relates to your chosen topic; a conclusion about your chosen topic and, possibly, some directions for future investigation.

Your report must be typed, single-spaced, 12pt font, 1in margins, between 5 and 10 pages in length (excluding references), and submitted in PDF format. Your report must be in clear English prose, utilizing proper spelling and grammar, and must include appropriately cited references.

The software artifact must be a novel and non-trivial application of the topic (at least 300 lines-of-code). The novelty is with respect to the source material; that is, your software should not simply reproduce examples given in a book, article, paper, or manual; your software must be your work. Of course, you may find inspiration for your software from examples. Another good source of inspiration is assignments and projects that you have done in other (imperative and/or object-oriented) languages for other courses.

Grading

Grades for the project will be assigned based on the following grading scheme:

<table>
<thead>
<tr>
<th>Component</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic</td>
<td>10%</td>
</tr>
<tr>
<td>Report</td>
<td>45%</td>
</tr>
<tr>
<td>Software</td>
<td>45%</td>
</tr>
</tbody>
</table>
Mid-term Exam

There will be one mid-term exam; see below for date.

The mid-term exam must be taken at its scheduled time. Make-up mid-term exams will not be administered, unless exceptional circumstances have been discussed with the instructor in advance of the exam date and/or other arrangements have been made.

Final Exam

There will be a final exam; see below for the date. The final will be comprehensive and will cover material from the entire course, including readings, lectures, and assignments.

The final exam must be taken at its scheduled time. Any exam conflicts must be reported to the instructor by the end of Week 6 (see the RIT Final Examination Policies).

Late Policy

Assignments that are submitted electronically (most assignments) will generally be due at 11:59PM on the due date. Assignments that are submitted in person will generally be due at the beginning of a class period.

Assignments are to be submitted on time. However, to accommodate the occasional difficulty with meeting an assignment due date, each student begins the term with five “extension tokens.” By spending an extension token, you will receive a 24-hour extension on a single assignment. To spend an extension token, you must e-mail the instructor before the assignment is due; you cannot spend an extension token after an assignment’s due date has passed. You may spend at most one extension token on a single assignment. After spending five extension tokens, late assignments will not be accepted.

Regrading

After a graded exam or assignment has been returned, you have one week to bring any questions about grading to the instructor’s attention. No grade adjustments will be made after this time.

Important Dates

April 8 (Mon.): Mid-term Exam (in class, 110min)
April 12 (Fri.): Project topic
May 6 (Mon.): Project report & software
May 14 (Tue.): Final Exam (8:00am – 10:00am; GOL-3455; Section 01)
Academic Integrity

As with all courses, the [RIT Honor Code](#) and the [RIT Academic Honesty Policy](#) apply. See the Department of Computer Science’s statement on academic integrity for more details.

In this course, all submitted work must be your own work (i.e., written or programmed by you alone, unless explicitly stated otherwise) and must include acknowledgments of any collaborators or sources (other than course text books or handouts) used to produce your submission.

You are encouraged to discuss course material with other students. Discussion of assignments is also allowed, but sharing solutions or code is not allowed.

Disclaimer

I reserve the right to make any changes to the syllabus as I deem necessary throughout the course. Minor changes, such as assignment due dates, will be announced orally during class and posted on the course mailing list and home page. Major changes, such as grading percentages, will additionally be provided in writing.
Haskell is a purely functional programming language that utilizes functions to construct programs. Utilizing Haskell functions is presumed to have no side effects, as the evaluation order of the functions is independent [10]. Modern functional languages are characterized by being strongly typed, concise, clear, lazy, and easy to ensure correctness. Introducing functional programming in the Haskell language, this book is written for students and programmers with little or no experience. It emphasizes the process of crafting programmes, problem solving and avoiding common programming pitfalls. Covering basic functional programming, through abstraction to larger scale programming, students are lead step by step through the basics, before being introduced to more advanced topics. This edition includes new material on testing and domain-specific languages and a variety of new examples and case studies, including simple games. The third edition of Haskell: The Craft of Functional Programming is essential reading for beginners to functional programming and newcomers to the Haskell programming language. The emphasis is on the process of crafting programs and the text contains many examples and running case studies, as well as advice on program design, testing, problem solving and how to avoid common pitfalls. Revisions to this new edition include new material on testing and domain-specific languages and a variety of new examples and case studies, including simple games. Existing material has been expanded and re-order