**Computer Science 154: Introduction to Functional Programming**  
*Fritz Ruehr • Willamette University • Fall 2012*

**Introduction**
This course provides a broad introduction to programming in the functional style, including motivations, history, programming techniques and theory. Functional programming provides concise and elegant solutions to many problems, using an approach based on mathematics, logic and proof. The course will be taught in Haskell, a powerful, modern programming language which can be used for both mathematical investigations and serious system development. Topics covered will include a broad introduction to computing, symbolic representation of data, list manipulation, recursion, algebraic data types, higher-order functions and type systems. The study of functional programming languages provides a useful foundation and perspective for further study of topics in algebra, logic, programming languages, computer science theory and linguistics.

**Instructor**

*Name:* Fritz Ruehr  
*Office:* 208 Ford Hall  
*Phone:* 370-6165  
*E-mail:* fruehr@willamette.edu

*Office hours:* Th 1:00-3:00; by appointment; or just drop by!

*Course homepage:* http://www.willamette.edu/~fruehr/154

**Logistics and attendance**

<table>
<thead>
<tr>
<th>Lectures</th>
<th>MWF 1:50-2:50</th>
<th>Ford 204</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lab</td>
<td>MWF 3pm-4:00pm</td>
<td>Ford 224 (Math Windows lab—East end)</td>
</tr>
</tbody>
</table>

Students are expected to attend all lectures and at least one lab session per week for help or to demo their labs. You are in any case responsible for all content, changes in assignments or policies, etc., which are made during the course of scheduled classes. I will try to make important announcements available in the News section of the course home page or by e-mail.

**Grading of programming projects will be based on an interactive, hands-on “demo:”** when you are confident that your program is complete and correct, you will seek me out (preferably during lab) and ask to show me your work. I will look over your running application and your written program, try different inputs and features, verify that your program is correct and well-designed, and giving you feedback as appropriate. If I find errors or problems that could be easily fixed, I will try to give you an opportunity to make changes before the due date. **We feel that this is a flexible and humane approach to the grading process which maximizes interaction between students and faculty.**

You may also be asked to complete some **written homework assignments**—these will be assigned in lecture and either collected and graded by the instructor or self-graded (in the latter case a solution key will be provided).

**Textbook**

Our textbook will be *Programming in Haskell*, by Graham Hutton of the University of Nottingham—it is available at the bookstore and via other vendors. ISBN: 0-521-69269-5.

You may wish to supplement the textbook with various tutorial materials available via the home page.
Grading policy

Grades will be based on programming projects, exams and class participation—in-class quizzes and written homework may also be included, as needed to determine student progress. Individual grades will be given in numeric form and then combined to determine an overall grade at the end of the semester, according to the weights listed below.

Much of the time you devote to class will be spent writing about 7-10 programming labs. I plan to give one shorter mid-term and a longer, comprehensive final (during the regularly scheduled exam period—see below). The weights used for the overall grade will be:

- 35-40% divided evenly among the programming projects;
- 15-20% total for in-class quizzes, written homework and class participation;
- 20% for the mid-term exam; and
- 25% for the final exam.

(See above regarding the “demo” style of program grading.)

Programming projects will generally be due one to two lectures (at the beginning of the course) or one to two weeks (toward the end of the course) after they are assigned. On occasion, class-wide extensions may be announced for various reasons. If you think you will be unlikely to be able to complete an assignment, contact me before the due date regarding the problem. Extensions may be granted in emergency situations, but not for those which can be anticipated in advance.

Note: the final exam is scheduled (by the College) for Saturday, Dec. 15, 2012, from 2-5 pm.

(Please make your summer travel plans accordingly.)

Collaboration and related issues

All programming projects, exams and other work you hand in should be your own. You are allowed (and encouraged) to seek help from other students for general study purposes, but you should never allow other people to do your work for you. You may use standard libraries supplied with Hugs or GHC, but you should leave all copyrights and attributions intact and clearly identify your own contributions. Violations of these rules will result in penalties according to College policies, but could a zero grade for the assignment or a failing grade for the course.

In essence, it is OK to discuss ideas with other students, but to copy a piece of writing or program from another source, or to have someone else dictate it verbatim, write it down or type it in for you is NOT allowed.

Documented disabilities accommodations

I will respect any accommodations authorized by the Office of Disabilities Services. Please tell me about these accommodations as soon as possible.
Tentative topic list

I hope to cover the following list of topics this semester at a rate of from 1 to 3 lectures per topic. Listen for classroom announcements or see the course homepage for any changes as the need develops.

- Introduction to the course
- Haskell programming environments
- Basic types and literals
- Functions and operators
- Basic structures (pairs, tuples, the Maybe type)
- List literals and Prelude functions
- Higher-order functions
- Value and function definition
- Algebraic data types and patterns
- Recursion
- Lazy evaluation and infinite data structures
- Type checking
- List comprehensions
- Type classes and instances
- Lambda notation
- Conal Elliott’s functional approach to graphics
- Binary and general trees
- Folds and unfolds
- Proving properties of programs
- Other applications  [*given time, and depending on student interest*]

Accreditation information

In accordance with new college accreditation initiatives, I am listing here the **student learning objectives** associated with this class (SLO numbers are relative to CS Department norms):

- **CS/SLO #2:** “Students will achieve proficiency in Computer Science skills (fundamentals of programming, computer organization, architecture, algorithms, theory, designing and implementing software).”
- **CS/SLO #5:** “Students will demonstrate the ability to work independently to analyze and solve problems.”