15-150 Course Missive

Spring, 2012

1 Course Staff

The course has two professors, three head teaching assistants, and 14 teaching assistants:

Instructors	Email
Daniel R. Licata	drl@cs.cmu.edu
Michael Erdmann	me@cs.cmu.edu

Head Teaching Assistants	Email
Ian Voysey	iev@andrew.cmu.edu
Luke Zarko	ltz@cs.cmu.edu
Michael Arntzenius	daekharel@gmail.com

Teaching Assistants	Email
Abby Motley	amotley@andrew.cmu.edu
Amy Zhang	amyz@andrew.cmu.edu
Brandon Bohrer	bbohrer@andrew.cmu.edu
Dan Yang	dsyang@andrew.cmu.edu
Eric Balkanski	ebalkans@andrew.cmu.edu
Esha Uboweja	euboweja@andrew.cmu.edu
Matt McKay	mjmckay@andrew.cmu.edu
Nick Kindberg	nkindber@andrew.cmu.edu
Rafee Memon	rmemon@andrew.cmu.edu
Rob Murcek	rmurcek@andrew.cmu.edu
Sri Raghavan	srikrish@andrew.cmu.edu
Laura Scharff	lscharff@andrew.cmu.edu
Tyler Hedrick	thedrick@andrew.cmu.edu
Todd Nowacki	tmn@andrew.cmu.edu

In general, you should mail the entire course staff by e-mailing 15-150tas@lists.andrew.cmu.edu, rather than mailing individual TAs. This will ensure a faster response.

2 Course Objectives

The purpose of this course is to introduce the theory and practice of *functional programming (FP)*. The characteristic feature of FP is the emphasis on computing by calculation. The traditional distinction between program and data characteristic of *imperative programming (IP)* is replaced by an emphasis on classifying expressions by *types* that specify their behavior. Types include familiar (fixed and arbitrary precision) numeric types, tuples and records (structs), classified values (objects), inductive types such as trees, functions with specified inputs and outputs, and commands such as input and output. Execution of imperative programs is generalized to evaluation of well-typed expressions by a smooth extension of mathematical experience to programming practice.

The advantages of FP are significant:

- *Verification*: There is a close correspondence between the reasoning that justifies the correctness of a program and the program itself. Principles of proof by induction go hand-in-hand with the programming technique of recursion.

- *Parallelism*: Evaluation of compound expressions is naturally parallel in that the values of subexpressions may be determined simultaneously without fear of interference or conflict among them. This gives rise to the central concepts of the work (sequential) and span (idealized parallel) complexity of a program, and allows programs to exploit available parallelism without fear of disrupting their correctness.

- *Abstraction*: FP stresses data-centric computation, with operations that act on compound data structures as whole, rather than via item-by-item processing. More generally, FP emphasizes the isolation of abstract types that clearly separate implementation from interface. Types are used to express and enforce abstraction boundaries, greatly enhancing maintainability of programs, and facilitating team development.

Moreover, FP generalizes IP by treating commands as forms of data that may be executed for their effects on the environment in which they occur.

Upon completion of this course, students will have acquired a mastery of basic functional programming technique, including the design of programs using types, the development of programs using mathematical verification techniques, and the exploitation of parallelism in applications. The specific skills we expect you to acquire are

1. to write functional programs
2. to analyze their sequential and parallel complexity
3. to reason mathematically about their correctness
4. to structure them using abstract types

Prerequisites 21-127: Concepts of Mathematics, or permission of instructor. Students will require some basic mathematical background, such as the ability to do a proof by mathematical induction, in order to reason about program correctness.

What's next? Completion of this course with a C or better is necessary and sufficient for entry into 15-210 Data Structures and Algorithms, which will build on the functional model of computation to develop a modern account of parallel algorithms for a wide variety of abstract types.

3 Activities and Assessments

The course activities will help you acquire these skills, and we will assess you based on how well you have done so. For example, we will grade your programs based not only on automated tests, but also on whether they adhere to the methodologies we teach—a correct program will not necessarily get a perfect score.

3.1 Academic Integrity

All students are expected to be familiar with, and to comply with, the University Policy on Cheating and Plagiarism. Any work submitted as a homework assignment or examination must be entirely your own and may not be derived from the work of others, whether a published or unpublished source, the worldwide web, another student, other textbooks, materials from another course (including prior semesters of this course), or any other person or program. You may not copy, examine, or alter anyone else's homework assignment or computer program, or use a computer program to transcribe or otherwise modify or copy anyone else's files. We may sometimes run automatic code comparison programs (such as MOSS). These programs are very good at detecting similarity between code, even code that has been purposefully obfuscated. Such programs can compare a submitted assignment against all other submitted assignments, against all known previous solutions of a problem, etc. The signal-to-noise ratio of such comparisons is usually very distinctive, making it very clear what code is a student's original creative work and what code is merely transcribed from some other source.

3.2 Lectures

Lectures will be held Tuesday and Thursday, noon-1:20pm (Doherty 2315). Lectures will be interactive, and sometimes include short in-class exercises. Participation and these exercises will count for a small percentage of your grade.

3.3 Lab

Each Wednesday, you will complete an 80-minute problem session. The intention of labs is for you to practice the skills introduced in lecture, under the supervision of the TAs, to prepare you for the week's homework. The lab assignment will be handed out on Tuesday. You are expected to familiarize yourself with the assignment and with related lecture material before lab.

Because different students work at different rates, we do not expect everyone to finish all of the problems in the lab handout. We do expect you to put in a good-faith effort to work on the provided problems and learn something. For guidance, the lab handout will identify how far we expect most students to get. However, credit will be determined by checking your work off with a TA. Labs will be graded on a scale of 0/1 scale: you will receive a 0 if you do not attend lab at all, you leave early, or you do not engage with the material (e.g. please check Twitter later). You will receive a 1 if you put in a good-faith effort and the TAs determine that you have made satisfactory progress.

Collaboration Policy The above paragraph on Academic Integrity does not apply to lab. Collaboration is strongly encouraged, and your lab TAs will be very active in providing immediate assistance. We will often ask you to do the labs in pairs, and you are free to ask other students for help, as well. Learn from each other!

3.4 Homeworks

For the most part, weekly homeworks will go out Tuesday after class and be due the following Wednesday at 9:00AM. Some homeworks will be longer (e.g. there will be two 2-week assignments) and others will be slightly off-schedule. See the course Web page for individual assignment times.

Homeworks are designed to help you internalize the course material; they consist of written problems and short to medium-sized programming tasks. Homework should be submitted electronically to your handin directory in the course Andrew AFS space. We ask that you typeset the written portions of assignments, as this can be a helpful opportunity to polish your answers. Other arrangements can be made by request (e.g. to combat repetitive-stress injuries).

Please ensure that your code hand-in compiles. If it does not, it will be subject to a uniform penalty of 20%. Similarly, if your PDF is invalid, it will not be printed and graded. We will provide check-scripts to validate your hand-ins; be sure to run them!

Collaboration Policy See the paragraph on Academic Integrity above. There is one exception to this paragraph for homeworks: to facilitate cooperative learning, it is permissible to discuss a homework assignment with other students, provided that the following whiteboard policy is respected. A discussion may take place at the whiteboard (or using scrap paper, etc.), but no one is allowed to take notes or record the discussion or what is written on the board, and you must allow four hours to lapse after any discussion before working on the assignment. The fact that you can recreate the solution from memory is taken as proof that you actually understood it.

Late Policy In past semesters, homeworks were due Monday night, and each student was given late days allowing them to occasionally continue to work until 9AM on Wednesday. To give you more flexibility in scheduling your work, we are trying a different policy this semester: we have extended the homework deadlines to Wednesday morning (previous semesters' late deadline) for all students on all assignments. **However, this deadline is final: no late homeworks will be accepted, except by special permission of the instructor.** We strongly encourage you to attempt to complete the homeworks by Monday nights. This way, when it takes longer than you expect,[1] you still have some time remaining.

3.5 Exams

The course will have two exams, a midterm, and a final exam during finals week. As the only completely non-collaborative type of assignment in the course, exams account for a significant portion of your grade. However, lest that sound too scary, the exams will not ask you to learn new skills or be very creative, just to demonstrate that you have acquired the skills we ask you to practice in homeworks and labs.

3.6 Letter Grades

We will compute an average of your scores on the assessments using the following weights:

In-lecture exercises	3%
Lab	7%
Homework	40%
Midterm	20%
Final exam	30%

To get an idea of how you are doing in the course, you can roughly think of each 10% as a letter grade ($90\% - 100\%$ is an A, $80\% - 90\%$ is a B, $70\% - 80\%$ is a C, $60\% - 70\%$ is a D, below that is

[1] Hofstadter's law: it always takes longer than you expect, even when you take into account Hofstadter's Law.

failing). However, this is **not** a guarantee: we may adjust the cut-offs, either upwards or downwards, to compensate for variation in the difficulties of the assessments. The final determination of letter grades is at the discretion of the instructors.

4 Where To Get Help

4.1 Office Hours

Professors' office hours will be held in their offices. TA office hour locations vary; see the course Web page. TAs will gladly clarify homework questions, explain concepts covered in programs and homeworks, help you with general questions about the class, and answer specific questions about homework problems that you are stuck on. However, the TAs will not do the assignment for you. If you come with a specific question, the TAs may give you a task to do/think about and ask you to come back later in the office hours.

4.2 Piazza and E-mail

We will use Piazza, a Web-based bulletin board, for public course announcements and discussions. You will get an e-mail at your Andrew account about accessing the course Piazza. Please use Piazza to ask public questions, but make sure they don't give away any answers or violate the academic integrity policy. If you have a question that is inappropriate to ask in public ("Here's my attempt at a solution; I'm stuck on. . . "), please come to TA hours or mail `15-150tas@lists.andrew.cmu.edu`.

4.3 The Web

Homeworks, labs, lecture notes, documentation, important notices, and nifty links will all be available from the course web page in a plethora of formats. Please check the web page as often as you can (within reason, of course—go out and enjoy the sunshine every now and again), as important notices and the like will be posted there. The web page can be found at `http://www.cs.cmu.edu/~15150/`.

4.4 Notes and Textbooks

This class has no required textbooks. Your main text will be the lecture notes available on the course web page. You are strongly encouraged to read the lecture notes from each class before the next one. We ask this of you because our course material builds on itself rather intensely, and we feel that mastering each lecture's material as it happens will be easier in the long run than falling behind and having to catch up. You are responsible for the material covered in class even if you do not attend or if it is not mentioned in the notes.

As supplementary materials, we will refer you to specific chapters of

> *Programming in Standard ML*, Robert Harper,
> `http://www.cs.cmu.edu/~rwh/smlbook/online.pdf`.