COMP 3007 22F Course Outline

Staff

Role	Name	Email
Instructor	Douglas Howe	douglashowe@cunet.carleton.ca
Head TA	Emma Sewell	emmasewell@cmail.carleton.ca
TA	Carter Brown	carterbrown@cmail.carleton.ca
TA	Youssef Zayed	mandardeshpande@cmail.carleton.ca

Prerequisites

The calendar prerequisites are COMP 2402 and COMP 1805. We won't be applying much of the specific content of those courses, but it's important to have proficiency in an imperative language (like Java) and an understanding of some basic mathematical foundations. Students would have gotten this from 2402 (and its prerequisites) and 1805.

Course learning outcomes

- basic competence in applying a modern functional programming language
- basic competence in using recursive techniques for problem solving and reasoning about recursive programs that operate on inductively-defined data-types
- an understanding of the different kinds of evaluation that can be used in functional languages, and basic competence applying programming techniques exploiting **lazy** computation
- a basic understanding of the relationship between, and relative merits of, functional vs imperative programming languages
- a basic understanding of the theoretical underpinnings of functional languages (the lambda-calculus)

- ability to provide and explain the applicability of selected advanced abstraction mechanisms from type theory
- familiarity with techniques for handling imperative operations, such as variable assignment, pointer manipulation, concurrency and exceptions, in a purely functional language
- an understanding of the concepts of borrowing and ownership as used in the Rust programming language, and how they are used to give memory-safety guarantees
- an ability to write simple efficient programs that involve pointers and respect the restrictions on pointer use imposed by Rust

Course structure

The instructor will add handwritten notes to partially typeset slides during the lecture. The lectures will also include live coding. The resulting slides and code will be posted on the course website along with the recording of the lecture. Lecture attendance is optional.

Extensive use will be made of "Ed Discussion", a new Q&A tool being used by most of the top US universities. This will be the place where you can ask questions and have them answered. Answers can be from other students or from course staff (instructor and TAs), and answers by students can be annotated/approved/highlighted by staff. It is expected that most interaction outside of class will be done this way, and that conventional 1-1 office hours will only only be for students who are struggling with the material enough that they are unable to formulate specific questions.

Almost all of the term work of the course will involve the Haskell programming languages (see below on course software). There will be small, roughly weekly, assignments with quick grading turnaround.

Three of the lecture slots will be used for proctored quizzes. There will be a final exam during the scheduled exam period. All tests will be handwritten.

Important Dates

Date	Event
Sept 8	first 3007 lecture
Sept 20	last day for registration changes
Oct 4	Quiz 1
Oct 24-28	fall break
Nov 10	Quiz 2
Nov 15	last day to withdraw
Dec 6	Quiz 3
Dec 8	last 3007 lecture

Assignments will usually be posted on Thursdays and will be due the following Thursday.

Grading scheme

Weight	Component
40%	Assignments: equally weighted; two non-submissions allowed without penalty
35%	Quizzes: equally weighted; one absence allowed without penalty
25%	Final exam: see Note 1 below

Notes

1. Getting **below 40%** on the final exam will automatically make the**final grade F regardless of term work**.
2. The "non-submissions" and "absence" above are not the same as "best of". You can miss one quiz without penalty, but if you write all three then all three will be used to compute the Quiz portion of your grade. The policy is meant to account for illness etc without the burden of providing evidence.

Course software

We will be using the language *Haskell* for this course. You can download Haskell from haskell.org. There are numerous IDES that support Haskell, but you don't really need one. A simple text editor, together with ghci, a read-eval-print loop that runs in a terminal, are all you need. If you are using Windows, you're best

off using WSL.

If you do want a full-featured programming environment, I strongly recommend VSCode. If you want to use this, be sure to check the box for "Haskell Language Server" when downloading Haskell; it's unchecked by default. Then install the VSCode extension that's simply called "Haskell".

Accommodations

This course follows Carleton's policies. For a description of the kinds of accommodations available, and how to request them, see Carleton's Academic Accommodations page.