

## COMP 3007 Fall 2023

### Staff

Role	Name	Email
Instructor	Douglas Howe	<a href="mailto:douglashowe@cunet.carleton.ca">douglashowe@cunet.carleton.ca</a>
TAs TBD		

### Prerequisites

COMP 1805 and COMP 2402. Prerequisites are enforced for this course. Those without the prerequisites will be deregistered.

### Laptop Requirement

All students in the course are required to have a laptop that they can bring to class. The lectures will not require them, but the in-class quizzes and final exam will.

### 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, 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 and exceptions, in a purely functional language

There may also be a section of the course using the Go programming language, with learning outcomes related to concurrency and “duck-typing”.

## Course structure

This is a *synchronous in-person* course. The instructor intends to record lectures and post them on the course website to assist students who are sick etc, but there is no guarantee. If for some reason a lecture does not get recorded, it will not be redone. Lectures might be streamed live, but this should *not* be counted on.

Lectures will be a mix of slides and “live coding”. The slides and code will be posted on the course website along with a recording of the lecture. Lecture attendance is optional (but see above about videos).

Extensive use will be made of “Ed Discussion”, a Q&A tool being used by most of the top US universities. This will be the main place where you can ask questions and have them answered. Answers can be from course staff (instructor and TAs), or from other students. Answers by students can be endorsed/annotated/highlighted by staff. It is expected that most interaction outside of class will be done this way. 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 or have so many questions that an in-person meeting is more efficient.

Almost all of the term work of the course will involve the Haskell programming languages (see below on course software). See the “Help” section of the course website for instructions on how install Haskell. )

## Textbook and references

There is no textbook for the course, but the online book [Learn you a Haskell for Great Good](#) is an excellent tutorial-style book for learning basic Haskell. Relevant sections will be pointed out at the course proceeds. Other web resources will be specified during the course.

## Discord vs Ed

There is no official Discord channel for the course. Students, as always, are free to use one, as long as they adhere to University rules about harassment etc, but the course staff will not be answering course-related questions there, and will not be correcting any misinformation appearing there. The Ed discussion tool is the place to get help with course material, and to discuss it with other students.

## Assessment scheme

Weight	Course component
10%	Weekly assignments, lowest two dropped
45%	4 quizzes @15%, lowest one dropped
45%	Final exam

The assessments will almost exclusively focus on coding skills and will involve writing small programs that will be tested by an automated grader.

## Late assignments

Late assignments might be graded, If they are, they will still receive a zero no matter what the reason for lateness. Any grade feedback would just be for information purposes.

## Missed assignments and quizzes

The dropping of low scores in the assessment scheme is intended to account for work that's missed for reasons beyond a student's control. There will be no other accommodations granted for missed quizzes or assignments *no matter what* the reason, so be careful save your "free passes" for illnesses or other events out of your control.

## E-proctoring

Quizzes will be done in-class using your laptop. You will write and test programs, and submit the code to Gradescope (a grading tool connected to Brightspace).

Unfortunately, for this to work in a classroom setting it's essential to use e-proctoring to make sure students aren't getting outside help.

The quizzes and the final exam will use CoMaS, an e-proctoring tool developed in the

Carleton's School of Computer Science and now officially supported by our Exam Services.

CoMaS is a small tool you install on your laptop for the exam period. It does not look at anything that was on your laptop before the exam starts. During the exam, it takes occasional screen shots and webcam pictures, and monitors file changes. After the exam the tool can be deleted.

The minimum system requirements for your laptop to run CoMaS are [here](#).

CoMaS has been thoroughly vetted by the university. If you're concerned about privacy, maybe the following will help.

1. You almost certainly have already installed apps from sources far less trustworthy than a large public university, including sources whose business model relies on harvesting personal information.
2. A breach of student privacy using CoMaS, if it became public, would be an existential crisis for the university. One thing I've learned about senior academic management is that if anything gives them nightmares it's the possibility of negative national press.

If you have questions about CoMaS and privacy, please see the [CoMaS Privacy FAQ](#).

If for some reason you can't tolerate temporarily installing CoMaS, the only alternative is to write the test on paper. You wouldn't get the help of a programming environment, or autograding in Gradescope to help with debugging, but it would still be a reasonably fair test. The programs you will be asked to write will be tiny.

## Tentative course schedule

Date	Event
Wed Sep 04	First lecture
Sun Sep 08	A1 due
Sun Sep 15	A2 due
Sun Sep 22	A3 due
Mon Sep 23	Quiz 1

Sun Sep 29	A4 due
Sun Oct 06	A5 due
Mon Oct 14	Thanksgiving, A6 due
Wed Oct 16	Quiz 2
Mon Oct 21	fall break
Wed Oct 23	fall break
Sun Oct 27	A7 due
Sun Nov 03	A8 due
Sun Nov 10	A9
Mon Nov 11	Quiz 3
Sun Nov 18	A10 due
Sun Nov 24	A11 due
Sun Dec 01	A12 due
Mon Dec 02	Quiz 4
Wed Dec 04	Last lecture

## Academic integrity

Current information on academic misconduct (definitions, consequences etc) can be found [here](#).

## Accommodations

The 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.

