# Computer Science 3MI3 - Principles of Programming Languages

2020 Course Outline

Mark Armstrong

August 2nd, 2020

## Contents

# 1 TL;DR (Too Long, Didn’t Read)

At the very least, please review these sections of the outline.

- Course staff
- Communicating with course staff
- Your responsibilities regarding course administration tools
- Asking well-posed questions
- Grading,
  - Missed work,
  - Homework due dates and
  - Assignment due dates.

# 2 The purpose of an outline

> “A course outline sets the expectations for students and what they can expect in terms of the course experience they will receive, the format in which the course will be delivered and the knowledge and skills that can be gained. The outline introduces the course and the instructor and sets out the expectations of

the instructor so that students are aware of how they will learn, what level of participation will be expected and how they will be assessed."

# 3 Course staff

## 3.1 Instructor: Mark Armstrong

- Email: armstmp@mcmaster.ca
- Website: `https://armkeh.github.io`

(Digital) office/student conference hours available on request.

### 3.2 Teaching assistant: Habib Ghaffari-Hadigheh

- Email: ghaffh1@mcmaster.ca
- Website: https://ghhabib.me/

(Digital) office/student conference hours available on request.

## 4 Schedule

| | Mon | Tues | Wed | Thu | Fri | Sat | Sun |
|---|---|---|---|---|---|---|---|
| 9:30 | | | Tutorial 2 | | | | |
| 10:30 | | | | | | | |
| 11:30 | Lecture | | Lecture | | | | |
| 12:30 | Tutorial 1 | | | | | | |
| 13:30 | | | | | Lecture | | |
| EOD | | | | | Homework released | | Homework due |

### 4.1 Lecture recordings

Lectures and tutorials will be made available as recordings shortly after said lectures and tutorials.

The platform for said recordings will be announced after the start of classes. :TODO:

### 4.2 Homework due dates

This course will have weekly homeworks, as described in Grading.

Homework will be released each Friday by end of day and will be due the Sunday nine days later by end of day, unless a delay is announced on the course homepage, and except for the weekends surrounding the midterm recess —there will be a single homework due the Monday following the recess instead.

### 4.3 Assignment due dates

This course will have three assignments, as described in Grading.

As of the beginning of the course, the assignments are planned to be due on

- October 7th,
- November 11th, and
- December 16th.

Any changes to these dates will be announced on the course homepage. Also see Accomodations for SAS and conflicts with other courses; in particular, we will reevaluate the final assignment due date once the exams schedule is released, based on the exam dates of other third year computer science courses.

## 5 Administration tools

### 5.1 The tools

This course will be administered via a combination of

- a "team" on the CAS departmental Microsoft Teams,
  - —Zoom meetings may be used if there is a problem with Teams; any such change will be announced on the homepage—
- the course homepage,
- a Github repository of the course content, from which the homepage is hosted as a `github.io` website,

- a repository for each student on the McMaster CAS GitLab server, and
- communications via McMaster email addresses.

Specifically,

**Teams** will be used for live lectures/tutorials, lecture/tutorial recordings, and preferred for discussions relevant to the whole (or at least many members of) the class,

**Zoom** may be used for live lectures/tutorials if there are problems with Teams,

**the homepage** will be used for announcements and convenient access to notes and homework/assignments,

**the Github repository** will be used to host the course homepage and content, and allow students to easily see version changes to content,

**the Gitlab repository for each student** will be used for homework and assignment submissions and grade distribution, and

**McMaster email addresses** will be used for private communications with students.

An Avenue to Learn course has been created for this course for the sake of directing students to the course homepage and entering homework/assignment deadlines in a calendar. No course content will be uploaded to Avenue to Learn, and attempts to communicate with staff on that platform may go unnoticed and unanswered.

### 5.2 Your responsibilities regarding course administration tools

**It is the student's responsibility**

- to ensure they have an account on the McMaster CAS GitLab server and the CAS Microsoft Teams team
- to be aware of the information on the course's homepage and
- to check the homepage, their course GitLab repository, the Microsoft Teams team for the course and their email regularly for announcements and changes.

It is not assumed that students follow the Github repo, but it is a good practice to stay informed of any and all changes to content.

# 6 Communicating with course staff

To communicate with course staff reliably, you should choose the most appropriate means from the below.

- "Mention" the course staff member in a relevant channel on the Microsoft Teams team.
  - This is appropriate for questions which may interest many students.
- Private message the course staff member on Microsoft Teams.
  - This is appropriate for very quick questions.
- Email the course staff member using the email listed under Course staff.
  - This is appropriate for longer or more detailed questions.

Note that outside of class hours, course staff may not be available for immediate replies to your communication. Permit up to a business day for response before following up on urgent issues, and up to two business days for non-urgent issues.

## 6.1 Asking well-posed questions

While there are "no stupid questions", and course staff are happy to help with any questions regarding course material, it is expected that when students ask questions they will take the time to ask well-posed questions as described in these resources.

- The computer science StackOverflow guide to asking good questions.
- Asking Technical Questions by Dr. Clarkson of Cornell University.
- How to Ask Questions The Smart Way by Eric Steven Raymond and Rick Moen.

In summary, when asking questions, always take the time to do your own research first, and describe this research to the staff.

- For questions regarding course administration, always check the homepage and the outline first, and include which sections you have checked in your question.

- For questions regarding tools, including installation and usage, always search online first, and list the resources consulted in your question.
- For questions regarding course material, always reference the portions of the notes you have checked for your answers.

Failure to follow these practices may result in terse answers from course staff, such as "Please check the course outline."

## 7 Resources

The course notes are intended to be self contained, but the recommended texts and several of the available resources are available free of charge, so you are encouraged to investigate them.

The primary recommended (not required) text is

- Pierce 2002 – Types and Programming Languages
  - Available through the McMaster library on ProQuest Ebook Central. You may view the whole text online, download up to 65 pages per day as a PDF, or borrow the whole text using Adobe Digital Editions.
  - Available for sale through the Campus Store.

ACM Digital Library citation:

> Benjamin C. Pierce. 2002. Types and Programming Languages (1st. ed.). The MIT Press.

From its abstract:

> A type system is a syntactic method for automatically checking the absence of certain erroneous behaviors by classifying program phrases according to the kinds of values they compute. The study of type systems – and of programming languages from a type-theoretic perspective – has important applications in software engineering, language design, high-performance compilers, and security. This text provides a comprehensive introduction both to type systems in computer science and to the basic theory of programming languages. The approach is pragmatic and operational; each new concept is motivated by programming examples and the more theoretical sections are driven by the needs of implementations

## 7.1 Additional textbooks

### 7.1.1 "SICP"; "The Wizard Book"

ACM Digital Library citation:

> Harold Abelson and Gerald J. Sussman. 1996. Structure and Interpretation of Computer Programs (2nd ed.). MIT Press, Cambridge, MA, USA.

From its abstract:

> With an analytical and rigorous approach to problem solving and programming techniques,this book is oriented toward engineering. Structure and Interpretation of Computer Programs emphasizes the central role played by different approaches to dealing with time in computational models. Its unique approach makes it appropriate for an introduction to computer science courses, as well as programming languages and program design.

`html` available through the MIT press and `pdf` available through GitHub.

### 7.1.2 "Van Roy & Haridi"

ACM Digital Library citation:

> Peter Van Roy and Seif Haridi. 2004. Concepts, Techniques, and Models of Computer Programming (1st ed.). The MIT Press.

From its abstract:

> The book presents all major programming paradigms in a uniform framework that shows their deep relationships and how and where to use them together.

`pdf` available through CiteSeer$^{X}$.

### 7.1.3 "Dowek"

ACM Digital Library citation:

> Gilles Dowek. 2009. Principles of Programming Languages. Springer Publishing Company, Incorporated.

From its abstract:

This book is an introduction to the principles around which these languages are organised: imperative constructions, functional constructions, reference, dynamic data types, objects and more.

`pdf` available through the McMaster library.

#### 7.1.4 "Sebesta"

ACM Digital Library citation:

Robert W. Sebesta. 2012. Concepts of Programming Languages (10th ed.). Pearson.

An encyclopedic text on the construction of programming languages.

#### 7.1.5 "Fernández"

ACM Digital Library citation:

1. Fernandez. 2004.

Programming Languages and Operational Semantics: An Introduction. King's College Publications.

An introductory text covering primarily operational semantics of a simple imperative and a simple functional language.

`pdf` available through the McMaster library.

## 8 Course description

Here we provide both formal and informal descriptions and goals for this course.

### 8.1 Calendar description

Design space of programming languages; abstraction and modularization concepts and mechanisms; programming in non-procedural (functional and logic) paradigms; introduction to programming language semantics.

## 8.2 Informal objectives

- Investigate a number of programming languages which exemplify different paradigms.
  - A relatively shallow but comprehensive survey.
  - Focusing on general-purpose languages.
- *Formally* describe programming language syntax and semantics.
  - An application of theory learned previously.
- Apply various abstraction and modularisation techniques,
  - Learning how to apply them and to which situations they are best applied.

## 8.3 Course preconditions

Before beginning this course:

1. Students should know and understand:
   (a) Basic concepts about integers, sets, functions, & relations.
   (b) Induction and recursion.
   (c) First order logic, axiomatic theories & simple proof techniques.
   (d) Regular expressions & context-free grammars.
   (e) Programming in imperative languages.
   (f) Basic concepts of functional programming languages.
2. Students should be able to:
   (a) Produce proofs involving quantifiers and/or induction.
   (b) Understand the meaning of a given axiomatic theory.
   (c) Construct regular sets & context-free languages.
   (d) Produce small to medium scale programs in imperative languages.
   (e) Produce small scale programs in functional languages.

### 8.4 Course postconditions

After completion of this course:

1. Students should know and understand:
   (a) Programming in functional languages.
   (b) Programming in logical languages.
   (c) Formal definitions of syntax & semantics for various simple programming languages.
   (d) Various abstraction & modularisation techniques employed in programming languages.
2. Students should be able to:
   (a) Reason about the design space of programming languages, in particular tradeoffs & design issues.
   (b) Produce formal descriptions of syntax & semantics from informal descriptions, identifying ambiguities.
   (c) Select appropriate abstraction & modularisation techniques for a given problem.
   (d) Produce tools for domain-specific languages in imperative, functional and logical languages.

### 8.5 Formal rubric for the course

| Topic | Below | Marginal | Meets | Exceeds |
|---|---|---|---|---|
| Familiarity with various programming languages | Shows some competence in procedural languages, but not languages from other paradigms | Shows competence in procedural languages and limited competence in languages from other paradigms | Achieves competence with the basic usage of various languages | Achieves competence with intermediate usage of various languages |
| Ability to identify and make use of abstraction, modularisation constructs | Cannot consistently identify such constructs | Identifies such constructs, but does not consistently make use of them when programming | Identifies such constructs and shows some ability to make use of them when programming | Identifies sucj constructs and shows mastery of them when programming |
| Ability to comprehend and produce formal descriptions of PL syntax | Unable or rarely able to comprehend given grammars; does not identify ambiguity or precedence rules | Comprehends given grammars, but produces grammars which are ambiguous or which do not correctly specify precedence | Makes only minor errors regarding precedence or ambiguity when reading or producing grammars | Consistently fully understands given grammars and produces correct grammars. |
| Ability to comprehend and produce operational semantics for simple PLs | Rarely or never comprehends such semantic descriptions | Usually comprehends such semantic descriptions, but cannot consistently produce them | Comprehends such semantic descriptions and produces them with only minor errors | Comprehends such semantic descriptions and produces them without errors |

## 9 Grading

The graded work for this course consists of

- weekly short homeworks, which may include
  - short written answers and
  - relatively small programming tasks.
- three larger programming assignments

– involving several “medium size” programming tasks each.

**There will not be midterm or final examinations for this course.**

Portions of your grade for both the homeworks and the assignments may be calculated by automated unit tests. A limited sample of such unit tests will be provided before the deadline, at least for the assignments.

Each student's final grade will be calculated using the scheme

| | |
|---|---|
| Homework | 25% |
| Assignment 1 | 20% |
| Assignment 2 | 25% |
| Assignment 3 | 30% |

### 9.1 Accomodations for SAS and conflicts with other courses

If you require accommodation regarding course work deadlines, either with regards to SAS or with regards to course work in other courses, please contact Mark no later than **two weeks** before the deadline of the relevant course work deadline.

### 9.2 Missed work

A student who would like to receive accommodation for missed academic work due to an absence needs to complete a McMaster Student Absence Form (MSAF) on-line at `http://www.mcmaster.ca/msaf/`. When the MSAF tool asks you for the party who should receive your request for accommodation, enter `armstmp@mcmaster.ca`. MSAFs sent to any other email address will be ignored.

**Students are reminded that they are expected to contact the instructor after filling out an MSAF**.

For this course, the accomodation for any missed work will be a 4 day extension.

**The missed work must still be submitted**.

## 10 Approved advisory statements

The following two pages cover topics and policies related to undergraduate course management. Please review them.

# COURSE OUTLINE – APPROVED ADVISORY STATEMENTS

## ACADEMIC INTEGRITY

You are expected to exhibit honesty and use ethical behaviour in all aspects of the learning process. Academic credentials you earn are rooted in principles of honesty and academic integrity. It is your responsibility to understand what constitutes academic dishonesty.

Academic dishonesty is to knowingly act or fail to act in a way that results or could result in unearned academic credit or advantage. This behaviour can result in serious consequences, e.g. the grade of zero on an assignment, loss of credit with a notation on the transcript (notation reads: "Grade of F assigned for academic dishonesty"), and/or suspension or expulsion from the university. For information on the various types of academic dishonesty please refer to the *Academic Integrity Policy*, located at https://secretariat.mcmaster.ca/university-policies-procedures-guidelines/

The following illustrates only three forms of academic dishonesty:

- plagiarism, e.g. the submission of work that is not one's own or for which other credit has been obtained.
- improper collaboration in group work.
- copying or using unauthorized aids in tests and examinations.

## AUTHENTICITY / PLAGIARISM DETECTION

*Some courses may* use a web-based service (Turnitin.com) to reveal authenticity and ownership of student submitted work. For courses using such software, students will be expected to submit their work electronically either directly to Turnitin.com or via an online learning platform (e.g. A2L, etc.) using plagiarism detection (a service supported by Turnitin.com) so it can be checked for academic dishonesty.

Students who do not wish their work to be submitted through the plagiarism detection software must inform the Instructor before the assignment is due. No penalty will be assigned to a student who does not submit work to the plagiarism detection software. All submitted work is subject to normal verification that standards of academic integrity have been upheld (e.g., on-line search, other software, etc.). For more details about McMaster's use of Turnitin.com please go to www.mcmaster.ca/academicintegrity.

## COURSES WITH AN ON-LINE ELEMENT

*Some courses may* use on-line elements (e.g. e-mail, Avenue to Learn (A2L), LearnLink, web pages, capa, Moodle, ThinkingCap, etc.). Students should be aware that, when they access the electronic components of a course using these elements, private information such as first and last names, user names for the McMaster e-mail accounts, and program affiliation may become apparent to all other students in the same course. The available information is dependent on the technology used. Continuation in a course that uses on-line elements will be deemed consent to this disclosure. If you have any questions or concerns about such disclosure please discuss this with the course instructor.

## ONLINE PROCTORING

Some courses may use online proctoring software for tests and exams. This software may require students to turn on their video camera, present identification, monitor and record their computer activities, and/or lock/restrict their browser or other applications/software during tests or exams. This software may be required to be installed before the test/exam begins.

## CONDUCT EXPECTATIONS

As a McMaster student, you have the right to experience, and the responsibility to demonstrate, respectful and dignified interactions within all of our living, learning and working communities. These expectations are described in the *Code of Student Rights & Responsibilities* (the "Code"). All students share the responsibility of maintaining a positive environment for the academic and personal growth of all McMaster community members, whether in person or online.

It is essential that students be mindful of their interactions online, as the Code remains in effect in virtual learning environments. The Code applies to any interactions that adversely affect, disrupt, or interfere with reasonable participation in University activities. Student disruptions or behaviours that interfere with university functions on online platforms (e.g. use of Avenue 2 Learn, WebEx or Zoom for delivery), will be taken very seriously and will be investigated. Outcomes may include restriction or removal of the involved students' access to these platforms.

## ACADEMIC ACCOMMODATION OF STUDENTS WITH DISABILITIES

Students with disabilities who require academic accommodation must contact Student Accessibility Services (SAS) at 905-525-9140 ext. 28652 or sas@mcmaster.ca to make arrangements with a Program Coordinator. For further information, consult McMaster University's *Academic Accommodation of Students with Disabilities* policy.

## REQUESTS FOR RELIEF FOR MISSED ACADEMIC TERM WORK

McMaster Student Absence Form (MSAF): In the event of an absence for medical or other reasons, students should review and follow the Academic Regulation in the Undergraduate Calendar "Requests for Relief for Missed Academic Term Work".

## ACADEMIC ACCOMMODATION FOR RELIGIOUS, INDIGENOUS OR SPIRITUAL OBSERVANCES (RISO)

Students requiring academic accommodation based on religious, indigenous or spiritual observances should follow the procedures set out in the RISO policy. Students should submit their request to their Faculty Office ***normally within 10 working days*** of the beginning of term in which they anticipate a need for accommodation or to the Registrar's Office prior to their examinations. Students should also contact their instructors as soon as possible to make alternative arrangements for classes, assignments, and tests.

## COPYRIGHT AND RECORDING

Students are advised that lectures, demonstrations, performances, and any other course material provided by an instructor include copyright protected works. The Copyright Act and copyright law protect every original literary, dramatic, musical and artistic work, including lectures by University instructors

The recording of lectures, tutorials, or other methods of instruction may occur during a course. Recording may be done by either the instructor for the purpose of authorized distribution, or by a student for the purpose of personal study. Students should be aware that their voice and/or image may be recorded by others during the class. Please speak with the instructor if this is a concern for you.

## EXTREME CIRCUMSTANCES

The University reserves the right to change the dates and deadlines for any or all courses in extreme circumstances (e.g., severe weather, labour disruptions, etc.). Changes will be communicated through regular McMaster communication channels, such as McMaster Daily News, A2L and/or McMaster email.