Computer Science 3MI3 - Principles of Programming Languages

2019 Course Outline

Mark Armstrong

October 31, 2019

Contents

1 The purpose of this 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.

2 Course staff

2.1 Instructor: Mark Armstrong

- Email: armstmp “at” mcmaster.ca

- Website: http://www.cas.mcmaster.ca/~armstmp/
- Office: ITB-229

2.2 Teaching assistant: Musa Alhassy

- Email: alhassm “at” mcmaster.ca
- Website: https://alhassy.github.io
- Office: ITB-229

2.3 Teaching assistant: Maryam Kord

- Email: kordm “at” mcmaster.ca

3 Schedule

	Mon	Tues	Wed	Thu	Fri
9:30				Tutorial 2 ABB 270	
10:30	Lecture PC 155	Tutorial 1 BSB B155	Lecture PC 155	Lecture PC 155	

4 Administration tools

This course will be administered via a combination of

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

It is the student's responsibility

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

To communicate with course staff, you should use the emails listed under 2. Other means of communication may not be monitored (in particular, Avenue to Learn email addresses will not be monitored).

4.1 Disclosure of information

Students should be aware that, when they access the electronic components of this course, private information such as first and last names, user names for the McMaster email 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 this course will be deemed consent to this disclosure. If you have any questions or concerns about such disclosure please discuss this with the instructor.

5 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.

5.1 Recommended texts

These textbooks should be available at the Campus Bookstore. They are additionally available online at the included links.

5.1.1 "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}.

5.1.2 "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.

5.2 Additional resources

These texts may be of interest, or assist if something is unclear in the notes and recommended texts.

5.2.1 "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.

5.2.2 "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.

5.2.3 "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.

6 Purpose and goals of this course

The undergraduate calendar description of this course is

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

But we should be more specific.

6.1 Informal objectives

- Expose you to several programming languages.
  - A relatively shallow but comprehensive survey.
  - Focussing on general-purpose languages.
- *Formally* describe programming language syntax and semantics.
  - An application of theory you have learned previously.
- Analyse the building blocks of languages.
  - Programs can be *massive* entities, but they are composed of relatively few *small* pieces.
- Provide you with criteria by which to judge languages.
  - So you can identify what languages fit what tasks.
- Examine the origins of certain languages/groups of languages.
  - Historical context provides insight into why languages are designed the way they are.

6.2 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 language
   (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.

6.3 Course postconditions

After completion of this course:

1. Students should know and understand:
   (a) The basics of several programming languages.
   (b) Formal definitions of syntax & semantics for various simple programming languages.
   (c) 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 (relatively simple) programs in various languages, including languages from non-procedural paradigms.

6.4 Formal rubric for the course

Topic	Below	Marginal	Meets	Exceeds
Familiarity with various programming languages (PLs)	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
Ability to comprehend denotational and axiomatic semantics for simple PLs	Rarely or never comprehends such semantic descriptions	Inconsistently comprehends such semantic descriptions	Consistently comprehends such semantic descriptions	Consistently comprehends and can produce some simple semantic descriptions

7 Grading

The graded work for this course is:

- Weekly short homeworks, marked for completion,
  - (each student is permitted to miss 2 homeworks without penalty)
- ~~four~~ programming/written answer assignments,
  - (Updated as of October 15th, 2019)
- two midterm examinations, and
- the final examination.

Each student's final grade will be the maximum grade among three grading schemes.

	Scheme 1	Scheme 2	Scheme 3
Homework	10%	10%	10%
Assignments	3 * 10% = 30%	3 * 10% = 30%	3 * 10% = 30%
Midterm 1	10%	20%	10%
Midterm 2	20%	10%	10%
Final Exam	30%	30%	40%

In this way, your best examination is weighted more heavily.

8 Additional statements/information

8.1 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 a missed assignment will be a 4 day extension, and
- for a missed midterm will be a reallocation of the midterm/final examination weights to
  - 20% for the remaining midterm examination and

– 40% for the final examination.

No accomodation will be given for missed homework, as each student is already permitted to miss 2 homeworks without penalty.

In the case that further accomodations are required, the instructor reserves the right to conduct make-up oral examinations.

8.2 Academic integrity

You are expected to exhibit honesty and use ethical behavior in all aspects of the learning process. Academic credentials you earn are rooted in principles of honesty and academic integrity.

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 behavior 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.

It is your responsibility to understand what constitutes academic dishonesty. For information on the various types of academic dishonesty please refer to the Academic Integrity Policy, located at http://www.mcmaster.ca/academicintegrity/.

The following illustrates only three forms of academic dishonesty:

1. Plagiarism, e.g., the submission of work that is not one's own or for which other credit has been obtained.

2. Improper collaboration in group work.

3. Copying or using unauthorized aids in tests and examinations.

Your work must be your own. Plagiarism and copying will not be tolerated! If it is discovered that you plagiarized or copied, it will be considered as academic dishonesty.

Students may be asked to defend their written work orally.

8.3 Discrimination

The Faculty of Engineering is concerned with ensuring an environment that is free of all adverse discrimination. If there is a problem that cannot be resolved by discussion among the persons concerned, individuals are reminded that they should contact their Department Chair and the Human Rights and Equity Services (HRES) office as soon as possible.

8.4 Academic accomodation

Students who require academic accommodation must contact Student Accessibility Services (SAS) to make arrangements with a Program Coordinator. Academic accommodations must be arranged for each term of study. Student Accessibility Services can be contacted by phone 905-525-9140 ext.~28652 or email mailto:sas@mcmaster.ca. For further information, consult McMaster University's Policy for Academic Accommodation of Students with Disabilities.

Students requiring academic accommodation based on religious, indigenous on spiritual observances should follow the procedures set out in the RISO policy. Students requiring a RISO accommodation 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.

8.5 Course modifications

The instructor and university reserve the right to modify elements of the course during the term. The university may change the dates and deadlines for any or all courses in extreme circumstances. If either type of modification becomes necessary, reasonable notice and communication with the students will be given with explanation and the opportunity to comment on changes. It is the responsibility of the student to check their McMaster email and course web sites weekly during the term and to note any changes. Your McMaster email is the one with the address ending in `@mcmaster.ca`. This is a separate email address from your Avenue address.