UNIVERSITÉ Concordia UNIVERSITY

ACADEMIC YEAR: 2018-2019

Course Number COMP348	**Course Title** Principles of Programming Languages

Instructors

Section DD Kaustubha Mendhurwar (Coordinator)	**Office Hours** Wed 16:30-17:30 or Appointment via email	**Email** kaustubha.mendhurwar@concordia.ca **Office** EV3.231
Section U Abdelwahab Elnaka	**Office Hours** Wed 13:30-14:30 or Appointment via email	**Email** Abdelwahab.elnaka@concordia.ca **Office** EV 3.233
Section W Bahareh Goodarzi	**Office Hours** Thu 13:00 -14:00 or Appointment via email	**Email** bahareh.goodarzi@concordia.ca **Office** EV3.231

Teaching Assistants

1.	Basant Singh	mr.singh.basant@gmail.com
2.	Rashid Khokhar	rashidhk.engr@gmail.com
3.	Satnam Singh	s.satnam@outlook.com
4.	Ahmed Bataineh	ahmed_bataineh87@yahoo.com
5.	Himanshu Mahajan	himanshumahajan38@gmail.com

Schedule

DD 1	TBA	TBA	POD	Basant Singh	TBA
DD 1	TBA	TBA	POD	Rashid Khokhar	TBA
DD 1	TBA	TBA	POD	Satnam Singh	TBA
DDDA1	-T------	14:45 to 15:35	Tutorial	Ahmed Bataineh	MB S1.105
DDDB1	---J-----	14:45 to 15:35	Tutorial	Ahmed Bataineh	FB 620
DDDC1	-T------	14:45 to 15:35	Tutorial	Satnam Singh	H 529
DDDD1	---J----	15:45 to 16:35	Tutorial	Ahmed Bataineh	MB 2.265

Please consult the room numbers at the below mentioned page in case of any change:
https://www.concordia.ca/encs/students/course-schedules/fall-2018-2019.html

COURSE PREREQUISITES

COMP 249

COURSE DESCRIPTION

The objective of this course is to teach the principles of programming languages and basic skills needed to understand analyze and develop algorithms for problem solving using different paradigms. Examples from mathematics and data structures will be given and discussed. The following programming paradigms will be covered: logic, functional, procedural, imperative, object oriented programming, and Aspect-Oriented Programming.

COURSE CALENDAR DESCRIPTION

This course will cover the following topics:
Survey of programming paradigms: logic functional, Procedural, and Imperative programming. Issues in the design and implementation of programming languages. Declaration models: binding, visibility, and scope. Type systems, including static and dynamic typing. Parameter passing mechanisms. Hybrid language design.

LEARNING OUTCOMES

At the end of this course you should be able to:

- Understand the differences between different programming paradigms.
- Analyze problems and develop algorithms for problem solving.
- Develop programs in different programming languages

WEB PAGE

Lecture notes, assignments, and other course information and materials are either available via instructor course web page or Moodle (MyConcordia Portal)

TEACHING METHOD

The course comprises of weekly lectures and practical training; both in the form of tutorials and individual assignments. ***It is emphasized that the attendance in lectures as well as tutorials is mandatory for learning and performing well in this course.***

TUTORIALS

The tutorials will reinforce the material seen during the lectures with examples and exercises. The tutorials are also designed in a way that would assist you to get deeper knowledge and experience of problem analysis and problem solving. It is hence strongly recommended that you attend all the tutorials. Tutorials begin on the second week of classes.

POD (Programmer on Duty)

A POD is available weekly in the Computer Lab to answer course-programming questions. Times will be announced.

RECOMMENDED TEXTBOOKS

- Think Principles of Programming Languages, 3rd Edition, 2015. Course pack by Dr. C. Constantinides. This course pack covers the main course material (recommended).
- Learning Ruby, 2007. Michael Fitzgerald, O'Reilly
- Programming Language Pragmatics, 2009, Michael L. Scott, Elsevier

COURSE SCHEDULE

The list below provides a summary of the material that will be covered in this course as well as a tentative schedule. Tutorials will support the topics covered in the lectures and will provide hands-on exercises.

Weeks	Topics	Notes
Week 1	Course Overview and Introduction	
Week 2	Logic Programming with Prolog	
Week 3	Logic Programming with Prolog Continued	
Week 4	Logic Programming with Prolog Continued	
Week 5	Functional Programming with LISP	
Week 6	Functional Programming with LISP Continued	
Week 7	Functional Programming with LISP Continued	**Midterm Exam**
Week 8	Object Programming with Ruby	
Week 9	Object Programming with Ruby Continued	
Week 10	Procedural programming with C	
Week 11	Procedural programming with C Continued	
Week 12	Aspect-J	
Week 13	Aspect-J Continued	
Week 14	Object Programming with Java	
	Final Exam	**During exam period**

Disability requests: Please let your instructor know if you have a disability which requires special arrangements.

GRADING SCHEME

Components	%
Assignments (x3)	3 x 5% = 15%
Midterm	30%
Final Exam	55%

1. There is **no fixed**, a priori relationship between the numerical percentage and the final letter grades for this course.

2. To pass the course, students must have a passing mark in the final exam, as well as a passing total mark. Usually a score of 50% is required. **There are no make-ups/alternates for missed exams or assignments**.

DETAILS OF COURSE COMPONENTS

Exams: Both the exams will be closed book and will comprise of both multiple choice as well as descriptive questions.

The midterm exam will take place in week # 7 or 8 and the final exam during the exam period.

Assignments: There will be 3 assignments, which will be available online. These assignments play a major role in your learning of the various topics covered in this course. Assignments must be submitted on the due date through the course web page or Moodle site of the course (please check your section). **No late assignment is accepted.**

Submission: The assignments will have to be submitted by 23h55 on their respective due dates. Assignments submitted within the 3 days following the initial deadline will also be accepted for grading however, there will be a flat deduction of 20% and [if applicable] no bonus.

Attendance: Students are encouraged to attend the lectures as well as tutorials regularly, and are responsible for all the materials presented, discussed, and assigned.

Plagiarism: The most common offense under the Academic Code of Conduct is plagiarism which the Code defines as **"the presentation of the work of another person as one's own or without proper acknowledgement."**

This could be

- Material copied word for word from books, journals, internet sites, course notes, etc.
- Material that is paraphrased but closely resembles the original source.
- Work of a fellow student, e.g., an answer on a quiz, data for a lab report, a paper or
- Assignment completed by another student.
- A solution or a code purchased through one of the many available sources.

Plagiarism does not refer to words alone - it can also refer to copying images, graphs, tables, and ideas. "Presentation" is not limited to written work. It also includes oral presentations, computer assignments and artistic works. Finally, if you translate the work of another person into French or English and do not cite the source, this is also plagiarism.

In Simple Words: ***Do not copy, paraphrase or translate anything from anywhere without saying where you obtained it!***
(Source: The Academic Integrity Website:http://provost.concordia.ca/academicintegrity/plagiarism/)

CEAB/CIPS GRADUATE ATTRIBUTES

As part of both Computer Science and Software Engineering program curriculum, the content of this course includes material and exercises related to the teaching and evaluation of graduate attributes. Graduate attributes are skills that have been identified by the Canadian Engineering Accreditation Board (CEAB) and the Canadian Information Processing Society (CIPS) as being central to the formation of Engineers, Computer Scientists and IT professionals. As such, the accreditation criteria for the Software Engineering and Computer Science programs dictate that graduate attributes are taught and evaluated as part of the courses. COMP348 course aims at teaching and evaluating the following graduate attributes:

- **Demonstrate Knowledge:** Knowledge of the imperative as well as declarative programming paradigms with a wide array of programming technologies such as object-oriented, aspect-oriented, procedural, functional, functional object-oriented, logic programming and multi-paradigm programming. Clauses, procedures and queries, backtracking, the inferencing process, unification, instantiation and resolution. Expressions and lists, binding, defining functions, pure vs. side effecting functions, referential transparency, idempotence, higher-order and anonymous functions and function composition. Primitive vs. composite data types, pointers, arrays, records, unions and enumerated data types. Message passing object-oriented model, inheritance, polymorphism, static vs. dynamic object types, distinguishing between compiler and run-time system responsibilities, casting. Modularity, separation of concerns, crosscutting concerns, join points, point cut designators and advice. Introducing state and behavior into primary units of modularity, context passing, privileged and abstract aspects. Object extensions, modules, and Mixins. Introspection. Generic function object-oriented model, primary vs. auxiliary methods.
- **Problem Analysis:** Analyze problems to determine viable paradigm and language spaces and tradeoffs. Map solutions to specifications. Being able to move from specification to code. Analyze implementations to determine what task they perform and how they do it. Being able to move from code to specification.
- **Use of Engineering Tools:** Deployment of compilers, interpreters, integrated development environments (IDEs) for the implementation and debugging of programs. Being able to extend the functionality of IDEs with plugging. Use of application programming interfaces (APIs) as building blocks, use of libraries to support reuse. Use of frameworks for testing.

Please note: In the event of extraordinary circumstances beyond the University's control, the content and/or evaluation scheme in this course is subject to change.