CHM 411 Organic Chemistry IV 3-3-0
Selected advanced topics in organic chemistry are introduced; pericyclic reactions, organometallic chemistry; radical chemistry; and/or stereoselective reactions.
Pre-requisite: CHM 311

CHM 491 Independent Study 3-0-0

CHM 492 Independent Study 3-0-0

CHM 499 Honours Chemistry Research Project 6-0-12
Under the guidance of a faculty member, the student does an experimental research project requiring approximately 12 hours per week in both the Fall and Winter semesters and presents the results of the project in a seminar and a written dissertation. The project chosen must be approved in advance by the Department and maybe in any field of chemistry plus material science.
Prerequisites: Third Year Honours Chemistry registration or permission of the Department.

Computer Science

Faculty

Madjid Allili,
B.Sc.(Algiers), M.Sc., Ph.D.(Sherbrooke);
Professor

Layachi Bentabet,
B.Sc.(Eng.National Polytechnic, Algeria),
M.Sc.(Elec.Eng. Institut national des sciences appliquées, Lyon),
Ph.D.(Sherbrooke); Professor

Stefan D. Bruda,
B.Sc.Eng., M.Sc., Ph.D. (Queen's);
Professor

Russell Butler
B.Sc. (Bishop's), M.Sc., Ph.D. (Sherbrooke);
Associate Professor
Chair of the Department

Rachid Hedjam
B. Eng.(University of Setif, Algeria), M.Sc. (Montreal),
Ph.D. (ÉTS);
Assistant Professor

Yasir Malik,
M.Sc. (Ajou University, S. Korea), Ph.D. (Sherbrooke);
Associate Professor

Program Overview

Computer Science is the study of how abstract ideas are made to run and halt on a physical machine. The Bishop's Computer Science program develops this understanding progressively, grounding students in algorithms, data structures, and computer organization before advancing to the design and analysis of complex computing systems, including security-critical and intelligent systems. Students pursue focused pathways, in Theory, AI, or Cybersecurity, completing advanced applied or research work under close faculty supervision, culminating in a year-long capstone or honours thesis.

The department offers a range of programs, from broad foundational study to focused specialization:

1.) Undergraduate Programs
- B.Sc. Honours in Computer Science
- B.Sc. Major in Computer Science

2.) Multidisciplinary Program
- B.A. with a Major in Information Technology (BAIT)

3.) Additional Options
- Minor in Computer Science
- Certificate in Computer Science

The department also offers a Master's degree in Computer Science *(see Graduate Studies section of calendar).*

Undergraduate B.Sc. Degree Programs

Honours in Computer Science (120 credits) HONCSC

A. Dissertation-Based Honours

18 credits: Program prerequisites *(please refer to Table II in the Faculty section of the Calendar)*
57 CS credits: 36 required: CS 201, CS 211, CS 216, CS 304, CS 310, CS 311, CS 317, CS 321, CS 403, CS 409, CS 499
21 electives: must include 12 credits from 400-level courses and above
12 MAT credits: 12 required: MAT 108, MAT 200, MAT 206, MAT 207
3 PHY credits: PHY 101
3 credits: Humanities and Social Sciences requirement *(please refer to the Faculty section of the Calendar)*
27 credits of free electives

B.Course-Based Honours

The course-based stream does not require a dissertation (i.e. CS 499) but requires 3 more CS courses. It is primarily designed for students wishing a specialization in Computer Science but are not interested in research and do not intend to pursue graduate studies:

18 credits: Program prerequisites *(please refer to Table II in the Faculty section of the Calendar)*
60 CS credits: 30 required: CS 201, CS 211, CS 216, CS 304, CS 310, CS 311, CS 317, CS 321, CS 403, CS 409
30 electives: must include 15 credits from 400-level courses
12 MAT credits: 12 required: MAT 108, MAT 200, MAT 206, MAT 207
3 PHY credits: PHY 101
3 credits: Humanities and Social Sciences requirement *(please refer to the Faculty section of the Calendar)*
24 credits of free electives

General Notes for Honours

- after a minimum of 1 semester, a student with a grade of at least 80% in required courses may request entry to the Honours program
- the dissertation stream requires, in addition, departmental permission
- students must maintain an average of 80% in required courses to stay in the program

Computer Science Major (120 credits) MAJCSC

18 credits: Program prerequisites *(please refer to Table II in the Faculty section of the Calendar)*
45 CS credits: 30 required: CS 201, CS 211, CS 216, CS 304, CS 310, CS 311, CS 317, CS 321, CS 403, CS 409
15 electives
9 MAT credits: 6 required: MAT 108, MAT 200, 3 elective (MAT 19X cannot count as MAT elective)
3 PHY credits: PHY 101
3 credits: Humanities and Social Sciences requirement *(refer to the Faculty section of the Calendar)*
42 credits of free electives

Information Technology Major (B.A.) MAJITC

Information Technology (IT) is defined by the Information Technology Association of America (ITAA), as the study, design, development, implementation, support or management of computer-based information systems, particularly software applications and computer hardware. IT deals with the use of electronic computers and computer software to convert, store, protect, manage, transmit and retrieve data, securely.

This program provides the necessary skills and knowledge to work/design/participate within organizations that manage large amount of data and provide services to a large number of users. Students will develop skills and knowledge in Information Technologies, Management practices and Organizations, with the required fundamentals of Computer Science.

Note: Students following this degree program are not eligible to add a Business program

Program prerequisites: (12 credits)
Please refer to Table II in the Faculty section of the Calendar.

Core curriculum (36 credits):
CS 201, CS 211, CS214, CS 325, CS 304, CS 307
BCS 220, BCS 320, BCS 313, BCS 422,
BMA 140, BMG 100

Secondary Core [1] (30 credits)
A minimum of 3 courses in Computer Science.
A minimum of 3 courses in Business, normally chosen from the following list:
BCS 317, BCS 416, BCS 340, BCS 420,
BCS 430, BCS 450, BMG 214, BMG 323,
BMK 211, BMK 214, BMK 321, BMK 323,
BMK 333

[1] Students are advised to consult the Calendar for prerequisites

Arts and Science requirements (3 credits)
Please refer to the Faculty section of the Calendar

Free electives (39 credits)

Co-operative Education Program

B.Sc. Coop – PROCOP

The Co-operative Education Program combines a student's academic program with integrated work experiences through full-time work terms and regular academic sessions. The work terms are designed to present the students with the opportunity to blend theory and practice and to gain relevant work experience.

Each co-operative work term is between 12 and 16 weeks in length, and the student will be registered in a 3-credit Co-operative Placement course (CS 391, CS 392 or CS 393). These course credits count as free electives. Each is graded on a pass/fail basis and this grade is not included in the student's cumulative average. The evaluation is the responsibility of the Departmental Chair and will be based upon the submission of a work term report and a job performance report submitted by the employer. Normal academic regulations apply to the conduct and evaluation of the courses.

The number of work terms needed depends on the number of credits the students need to complete upon admission at Bishop's. Student who have been granted 30 advance credits (or more) will be required to complete two work terms (6 credits). Other students who have been admitted into a regular 120-credit degree program will be required to complete three work terms (9 credits). These credits will be added to the student's program and do not count as computer science courses, computer science electives. All work terms must be completed before the student's final academic semester and a student's last semester before graduation cannot be a work term. While every effort will be made to find a suitable placement for all students in the program, no guarantee of placement can be made since the employment process is competitive and subject to market conditions.

Admission to the Co-operative Education Program

Students must submit an application to be admitted to the program. Full-time students in any Honours or Major program offered in the Computer Science Department who have completed the online application package, who have successfully completed BMG191 and who have a minimum cumulative average of 70% upon application are admissible into the Co-op Program. Students in the Co-op must maintain their 70% average and be full-time in order to stay in the program.

Work Term Registration

Once a student has signed the Co-operative Education Agreement, the student may not drop the course associated with the work placement, except for exceptional circumstances. A student who decides to do so will not be able to stay in the Co-operative Education program.

Tuition and Fees

Each work term placement is a 3-credit course and students will pay tuition based upon their fee paying status (Quebec resident, Canadian out-of-province, International).

Work Term Evaluation

Successful completion of the work term is based upon the following:

- The receipt of a satisfactory job performance report from the employer
- The submission of a satisfactory work term report by the student.

The job performance report will be completed by the employer, using guidelines supplied by the Computer Science Department. It is the student's responsibility to ensure that the employer sends the completed evaluation to the Co-op Coordinator on or before the established deadline. Employer evaluations are confidential and are not reported on the student's transcript.

Computer Science Minor (24 credits)

MINCSC

9 required: CS 201, CS 211, CS 304
15 electives from any CS course

Certificate in Computer Science (30 credits)

CONCSC

Description and objectives:

The Certificate Program in Computer Science is designed for individuals who need to acquire a basic understanding of computers and programming and knowledge of the field in order to expand their area of interest and professional expertise. Topics include: Programming, Software Engineering, Web Design, Networks, Graphics, Artificial Intelligence and others. This program will help students to take full advantage of the computer technology available in the workplace.

Prerequisites to programs:

Applicants with insufficient Math background might be required to take an additional 3-credit Math course in their first semester (Math 190 or equivalent).

Program Overview

Certificate in Computer Science

12 required credits: CS 201, CS 211, CS 304, CS 321
18 credits of CS electives

General Notes/Restrictions:

1. Only one of CS 404, CS 408 or CS 499 may be taken for credit, unless with a special departmental authorization
2. Computer Science courses that are double-listed in Math cannot be counted toward fulfilling the Math electives required for the Computer Science Honours/Major.
3. Students must fulfill their Arts and Science requirements and Humanities requirements outlined in the Faculty section of the Calendar.

List of Courses

CS 201 Foundations of Computer Science 3-3-0
An introduction to Computer Science and selected applications suitable for both majors and science non-majors who want a broad overview of the field. The course provides a layered introduction covering algorithms, hardware, system software and applications. The course includes elementary programming using Python to illustrate the implementation of algorithms. Topics include Algorithmic foundations of Computer Science; The hardware world: number systems, Boolean logic, computer circuits, Von-Newmann architecture; System software: assembly language, and a choice of applications.
Note: Registration priority is given to Science and IT students. CS students must take this course in their first year. This course will use Grade 12 level mathematics. MAT 190 is a recommended prerequisite.

CSL 201 Foundations of Computer Science Laboratory 1-0-3
This is the practical laboratory for CS 201

CS 203 Full-Stack Web Development 3-3-0
Introductory course in web application development. Students learn how to design and build interactive websites that respond to user input and work with stored data. The course introduces the core technologies of the web, including HTML and CSS for page structure and visual design, JavaScript for basic interactivity, and simple server-side tools for handling requests and working with databases. Topics include how web pages communicate with a server, creating and processing forms, storing and retrieving information, basic user accounts, and an introduction to common web security considerations. Laboratory work emphasizes hands-on learning, with students incrementally building, testing, and deploying a small but complete web application.

CSL 203 Full-Stack Web Development Laboratory 1-0-3
Laboratory work focuses on the incremental development, testing, and deployment of a full-stack web application.

CS 208 Coding in Plain English 3-3-0
This course explores the limits of AI-assisted programming by treating natural language as a computational interface. Students design and test increasingly complex problems to probe what modern AI systems can and cannot do. Through structured experimentation, students prototype apps and games, solve domain-specific problems, and apply generative AI across disciplines. Emphasis is placed on experimental design, adversarial testing, validation, and critical analysis of results, positioning students to reason about AI systems not just as tools, but as evolving computational agents with measurable capabilities and limits.

CS 211 Introduction to Programming 3-3-0
This course introduces algorithms, data structures and software engineering principles. The use of a high level language is the tool to develop these components. By the end of the course, a successful student should be 'fluent' in programming, and have a good base for simple data structures. The course provides the necessary programming skills needed for further studies in Computer Science.

CSL 211 Introduction to Programming Laboratory 1-0-3
This is the practical laboratory for CS 211

CS 214 Introduction to Networks 3-3-0
This course covers the principles and protocols of modern computer networks. Topics include network models (OSI, TCP/IP), packet switching, addressing, routing algorithms, and programmable networking. Core Internet protocols (IP, TCP, UDP, DNS) are examined, along with network performance and reliability issues such as latency, throughput, congestion control, and fault tolerance. Laboratory work uses software-based emulation and simulation to analyze and design networked systems.

CS 216 Low-Level Programming 3-3-0
System programmers need to understand how a computer works at a low level. They program primarily in C, with some assembly language. This course covers number systems, the C programming language, and an assembly language for a representative processor architecture. Topics covered include addressing modes, the stack, function calls and argument passing.

CSL 216 Low-Level Programming Laboratory 1-0-3
Practical work for CS 216 will consist of programming in C and MIPS assembly language.

CS 219 General Topics in Computer Applications 3-3-0
The course will present general Computer Science-related topics, of interest to both Computer Science as well as non-Computer Science students. The course content is expected to vary to reflect the interest of students and Faculty, as well as market innovations.

CS 225 Foundations of Cybersecurity 3-3-0
This course introduces the fundamental principles of cybersecurity from a systems perspective. Students examine how security failures arise from interactions between software, hardware, networks, and humans, and learn to reason about threats, trust, and risk across computing systems. Topics include threat models, basic cryptography concepts, operating system and network security fundamentals, software vulnerabilities, hardware trust boundaries, and the human factors that shape real-world attacks. Emphasis is placed on understanding why systems fail, not just how attacks are executed.

CS 301 Computer Ethics 3-3-0
Technically minded professionals often give little attention to ethical issues. This course examines ethical, legal, and societal issues in modern computing, with emphasis on artificial intelligence, data privacy, and automated systems. Students apply ethical reasoning frameworks to topics such as algorithmic bias, surveillance, data ownership, automated decision-making, intellectual property, misinformation, and platform-scale social impacts. Issues of accountability, transparency, and responsibility in software and AI systems are explored through case studies and written analysis.

CS 304 Data Structures 3-3-0
This course examines the design, implementation, and analysis of fundamental data structures used in software systems, including arrays, linked lists, stacks, queues, trees, heaps, hash tables, and graphs. Emphasis is placed on algorithmic efficiency and rigorous time and space complexity analysis using asymptotic notation, with careful comparison of alternative data structure choices and their trade-offs. C++ is used as the implementation language to expose memory layout, pointer-based representations, and object semantics, allowing students to connect theoretical performance guarantees with runtime behavior.
Prerequisite: CS 211

CS 307 Using and Designing Data Bases 3-3-0
This course presents data modeling (Entity-Relationship model, UML, etc..), relational algebra, normalization, SQL language. Implementation of databases using the relational model is discussed. Object-oriented modeling and implementation is also introduced. Other topics include: Concurrency control, transaction processing, client-server systems, distributed databases, and web-based delivery of data.
Prerequisite: CS 304
Note: Students may not take this course for credit if they received credit for either BCS 214 (Jan 98 and onward) or CSC 274 (prior to 2003).

CSL 307 Using and Designing Data Bases Laboratory 1-0-3
This is the practical laboratory for CS 307

CS 308 Scientific Computing 3-3-0
This course introduces the principles and practice of scientific computing for modeling and simulation in science and engineering. Students learn how mathematical models are translated into numerical algorithms and implemented efficiently on modern computing systems. Topics include numerical methods for scientific problems, parallel algorithms for large-scale computation, performance modeling, and the impact of memory hierarchies on computational efficiency. Through hands-on work, students develop and analyze programs that run on multicore processors, clusters, and accelerators, gaining experience using high-performance computing techniques to solve real-world scientific and engineering problems at scale.
Prerequisites: CS 304, MAT 191, MAT 192
Note: See PHY 378. Students may not take this course for credit if they have received credit for MAT 279 or PHY 378.

CS 310 Introduction to Software Specifications 3-3-0
This course provides to all the students in CS degrees essential material on formal languages and automata, and also on program specification using logical predicates. The following topics will be addressed: introduction to techniques for specifying the behaviour of software, with applications of these techniques to design, verification, and construction of software; logic-based techniques such as loop invariants and class invariants; automata and grammar-based techniques, with applications to scanners, parsers, user-interface dialogs and embedded systems; computability issues in software specifications. These topics have been chosen because they are both theoretical and practical, and will be presented as such.
Prerequisite: CS 211
Prerequisite or Co-requisite: MAT 200

CS 311 Computer Organization and Logic Design 3-3-0
This is a theoretical course on computer organization and architecture. Different computer components and how they function are studied in detail. By the end of the course, students should be able to build (in theory) a small computer without interface. Topics covered are: boolean algebra and gates, combinational circuits (decoders, multiplexers, PLAs), logic design (flip-flops, shirts registers, counters, sequential circuits), the ALU, memory (RAM,ROM, secondary storage), I/O Devices and the control unit (hardwired, microprogramed). For those interested students, a follow-up course, largely consisting of lab experiments, is CS 312
Prerequisites: CS 201, CS 211

CSL 311 Computer Organization and Logic Design Laboratory 1-0-3
This is the practical laboratory for CS 311

CS 312 Embedded Systems 3-3-0
This course introduces embedded systems programming with an emphasis on the close integration of software and physical hardware. Students study how low-level software interacts with microcontroller architectures, peripherals, and real-time constraints. Topics include low-level programming, interrupts and timers, memory-mapped I/O, communication protocols (I²C, SPI, UART), real-time behavior, and hardware-oriented debugging. Practical understanding is developed through simulation-based exercises, code analysis, design problems, and project work that models real embedded systems such as autonomous devices and control-oriented applications.
Prerequisite: CS 216.

CS 316 Representation Learning 3-3-0
This course examines how structure is extracted from data through classical representation-learning methods grounded in linear algebra and optimization. Students study matrix-based models that uncover latent structure by optimizing explicit objectives under constraints such as low-rank structure, sparsity, orthogonality, and non-negativity. Emphasis is placed on understanding how these objectives shape the geometry of learned representations, how optimization procedures behave in practice, and when and why these methods succeed or fail. Through hands-on implementation from first principles, students analyze numerical stability, sensitivity to noise, and identifiability issues in widely used dimensionality reduction, factorization, and clustering techniques.
Prerequisites: CS 304 and MAT 200

CS 317 Design and Analysis of Algorithms 3-3-0
This course is intended to make students familiar with most of the existing techniques for problem solving. It starts with an introduction to algorithms efficiency, solving recurrence relations and basic data structures. Then different techniques for algorithms design are discussed; the divide-and-conquer technique, the greedy technique and its applications to graph algorithms, dynamic programming, backtracking and branch and bound algorithms. With every technique presented, examples from different domains are studied and their algorithms analyzed. At the end, students are briefly introduced to the vast area of "difficult" problems, or NP-complete.
Prerequisites: CS 304 and MAT 200

CS 321 Advanced Programming Techniques 3-3-0
The course is intended to be a sequel to introductory programming with emphasis placed on the architecture of software. It will go in depth into object-oriented techniques, reusability, data abstraction, class design, and implementation, design and structure of class libraries. Topics to be covered include: polymorphism, encapsulation, overloading, inheritance and delegation, types of inheritance (Inheritance for Extension, Specialization and Specification), composition, aggregation and design of collections. Static and dynamic types, downcasting, exception handling. The second half of the course will be devoted to software design patterns, with particular emphasis on the observer, iterator, visitor and selected creational patterns. Course work will involve significant programming projects. The teaching language will be Java.
Prerequisite: CS 304 Allow concurrent

CS 320 Natural Language Processing 3-3-0
This course provides a hands-on introduction to Natural Language Processing (NLP) using classical, non-neural approaches. Students will learn how to clean and normalize text, construct TF-IDF and bag-of-words representations, and train linear classifiers such as Logistic Regression, Naive Bayes, and Support Vector Machines. The course covers sequence labeling with Conditional Random Fields (CRF), topic modeling using Latent Dirichlet Allocation (LDA), information retrieval with BM25, and extractive summarization with TextRank. Students will also build n-gram language models with smoothing, evaluate models using precision, recall, F1, and perplexity, and discuss ethical issues including bias, privacy, and multilingual fairness. The emphasis is on practical implementation, evaluation, and error analysis using Python and standard NLP libraries.
Prerequisite: CS 304

CSL 321 Advanced Programming Techniques Laboratory 1-0-3
This is the practical laboratory for CS 321

CS 325 Computer & Network Security 3-3-0
This course provides an introduction to security and privacy issues in various aspects of computing, including cryptography, software, operating systems, networks, databases, and Internet applications. It examines causes of security and privacy breaches, and gives methods to help prevent them.
Prerequisite: CS 214

CS 330 Mobile App Development 3-3-0
This course covers Android mobile application development using Android Studio and Jetpack Compose. Topics include application architecture, state and lifecycle management, responsive UI design, performance constraints, and mobile software engineering patterns. Students work with event handling, navigation, local storage, device features, and external APIs. Testing, security fundamentals, deployment, and app store compliance are introduced. The course concludes with a final project in which students build and publish a functional mobile application.
Prerequisite: CS 211

CS 379 Electric Circuits and Electronics 3-3-3
Review of D.C. circuits, Kirchoff's laws, network theorems. Network analysis for A.C. circuits, phasors. Diode circuits and filters. The physical basis of semiconductor devices including semiconductor diodes, junction transistors, and field-effect transistors. The operation of transistor amplifiers, digital electronics and integrated circuits will also be covered.
Note: See PHY 319. Students may not take this course for credit if they have received credit for PHY 319.

CS 391 Co-operative Placement I 3-0-0
Students will integrate theory and practice through a related work placement
Prerequisite: admission to the Co-op Education Program

CS 392 Co-operative Placement II 3-0-0
Students will integrate theory and practice through related work placement
Prerequisite: CS 391

CS 393 Co-operative Placement III 3-0-0
Students will integrate theory and practice through related work placement
Prerequisite: CS 392

CS 400 Independent Studies 3-0-0
Individual study and research under the guidance of an advisor and Department staff.
Prerequisite: Permission of the department

CS 401 Embedded AI and Control 3-3-0
This course examines artificial intelligence deployed directly on embedded and cyber-physical systems where latency, reliability, safety, and security are non-negotiable. Students study how AI models are designed and integrated to operate locally without dependence on cloud connectivity. Topics include real-time inference, power and memory constraints, sensor fusion, control loops, scheduling and operating-system interaction, hardware accelerators, robustness and degraded-mode operation, security and privacy at the edge, validation and testing of rare failures, and safety-critical deployment in robotics, autonomous vehicles, industrial systems, and medical devices.
Prerequisite: CS 312

CS 402 Computer Graphics 3-3-0
This is an introductory course to the principles of interactive raster graphics. Topics include an introduction to basic graphics concepts, scan conversion techniques, 2-D and 3-D modeling and transformations, viewing transformations, projections, rendering techniques, graphical software packages and graphics systems. Students will use OpenGL graphics API to reinforce concepts and study fundamental computer graphics techniques.
Prerequisites: CS 304, MAT 108

CS 403 Principles of Programming Languages 3-3-0
This course provides a comparative study of the core concepts underlying programming languages and their implementations. Topics include language definition, syntax and semantics; compilation and interpretation; parsing techniques and parser construction; variables, binding, expressions, and control structures; data types and memory models; scope, procedures, and run-time systems; modularity and abstraction; concurrency, exception handling, and program correctness. The course also examines major programming paradigms, including functional, object-oriented, logic, and constraint-based programming.
Prerequisites: CS 304 and CS 310

CS 404 Project 3-0-3
This course is normally taken by CS students in their final year. The project must be approved in advance by the department. Students will be expected to submit a written report and to make a presentation.
Prerequisites: approval of the dept., 80% in CS courses

CS 405 Data Mining 3-3-0
Data is now created faster than humans are able to understand it and use it. There may be patterns hiding within this data with potentially useful information. This course will teach students, how to discover these patterns for the purpose of solving problems, gaining knowledge, and making predictions. Topics covered in this course include data preparation, clustering, classification, association rules for mining and models combination. This course includes assignments and a final project where the students are required to perform mining on real datasets.
Prerequisite: PHY 101 (or equivalent)
Note: See PHY 374. Students may not take this course for credit if they have received credit for P HY 374.

CS 406 Program Execution and Isolation 3-3-0
This course examines how programs are translated, executed, and managed at runtime. Core topics include parsing and syntax-directed translation, intermediate representations, runtime memory and execution models, optimization, code generation, and error detection, with emphasis on how compilers and runtimes preserve program meaning across abstraction boundaries. These foundations are used to study how execution environments enforce isolation and where those guarantees break down, including privilege transitions, system calls, memory-safety and time-of-check/time-of-use errors, side-channel leakage, speculative execution effects, and the limits of container and virtual-machine isolation. Case studies examine historically significant and contemporary vulnerabilities such as Dirty COW and TOCTOU to illustrate failure modes in real systems.
*Prerequisite: CS 310***CS 408 Project II 3-0-3**
This course is normally taken in the final year of studies and may involve work on a theoretical topic or a practical implementation of a sizable software project. The topic must be approved in advance by the department. Students are expected to attend bi-weekly project meetings where they present and discuss their work. In addition, they will make a final presentation at the end of term and submit a report.
Prerequisites: approval of the dept., 80% in CS courses.

CS 409 Principles of Operating Systems 3-3-0
Basic concepts of computer hardware; program translation linking and loading; cooperating sequential processes; critical section problem, process synchronization primitives, parallel programming; introduction to multiprogramming; operating system nucleus; file systems; reliability and protection; system performance, measurement and evaluation. Memory Management. Paging and Virtual memory. Unix. Using and programming the Unix Shell, Unix implementation. Examination of the implementation of Unix clones Minix, Linux, Survey of state-of the-art operating systems. Distributed Systems, Communication and synchronization in distributed systems. Theoretical issues and implementation.
Prerequisite: CS 304

CS 410 Software Engineering 3-3-0
Software is an engineered product that requires planning, analysis, design, implementation, testing and maintenance. This course is a presentation of the techniques used in each step of the software product process. Topics: software requirements analysis and specifications; software design process, object oriented design; testing, reliability and maintenance. Students will be expected to work jointly on several large software projects.
Prerequisites: CS 304, CS 310, CS 321, CS 403 (allow concurrent)

CS 411 Advanced Computer Architecture 3-3-0
The focus in this course is on basic principles, current practice, and issues in computer architecture and organization. At the end of the course students will have gained an understanding of how a computing system is organized, as well as why it is organized this way. The relation between hardware and the software that runs on it is emphasized, leading to an intuitive understanding of how the behavior of applications influences computer organization and design. Topics covered typically include (but are not limited to): instruction set design, micro-programed versus hardwired processors, pipelining and superscalar processors, memory organization (cache, primary, virtual), I/O and interrupts, multiprocessors. Comparative critical and quantitative analyses of various systems that currently exist are presented.
Prerequisite: CS 311 or instructor's permission.

CS 412 Computer Games Design 3-3-0
This course will explore the theory and practice of video game design and programming. Students will learn the basic concepts and techniques for the design and development of digital games. The topics covered in this course will include the history and taxonomy of video games, the basic building blocks of a game, computer graphics and programming, use interface and interaction design, and the software architecture for video games. It is assumed that students have taken courses in programming (best if it includes C or C++) and data structures. A good background in algorithms and basic mathematics (matrix algebra, trigonometry, linear algebra, vector calculus) is an asset for this course.
Prerequisite: CS 304

CS 415 Distributed Systems 3-3-0
This course studies the principles and practice of distributed systems: collections of independent computers that cooperate to provide a unified service. We focus on the fundamental challenges of distribution—partial failure, concurrency, unreliable communication, and the absence of a global clock—and the abstractions used to manage them. Topics include communication models, time and ordering, consistency models, replication, fault tolerance, and consensus. Students will analyze classic impossibility results and core algorithms, and apply these ideas to the design of scalable, reliable systems. Emphasis is placed on reasoning about correctness, understanding trade-offs, and designing for failure. Through readings, problem sets, and programming projects, students will learn how modern distributed systems balance consistency, availability, and performance, and how theoretical limits shape real-world system design.
Prerequisites: CS 304, CS 310

CS 416 Special Topics in Cybersecurity 3-3-0
This course explores core topics in cybersecurity from both theoretical and practical perspectives. Students examine how security vulnerabilities arise across systems, networks, and applications, and how defensive strategies can be applied throughout the system lifecycle. Topics may include threat modeling, cryptography, network security, authentication and access control, system hardening, vulnerability analysis, incident response, and risk management. Emphasis is placed on understanding attack surfaces, adversarial thinking, and evaluating system resilience. Projects involve analyzing and securing complex computing environments.
Prerequisites: CS 410

CS 419 Advanced Low-Level Programming 3-3-0
This course examines how software brings up, controls, and extends complete computing systems at the hardware level. Students work with system-on-chip platforms to study boot processes, memory layout, interrupts, concurrency, and kernel–user space interaction, and to implement device drivers and hardware abstraction layers using memory-mapped I/O and inter-process communication. Development emphasizes cross-compilation, board support packages, and low-level debugging with tools such as gdb, JTAG, and system tracing, with projects focused on correctness, performance, and reliability on real hardware.
Prerequisites: CS 312

CS 420 Large Language Models 3-3-0
Cross-level listed with CS 520. This course introduces the architecture and practical use of modern Large Language Models. Topics include transformers, tokenization, datasets, prompt engineering, instruction tuning, parameter-efficient fine-tuning (e.g., LoRA), retrieval-augmented generation, evaluation, and deployment considerations. Students work hands-on with pre-trained models using Python and PyTorch to adapt LLMs to specific tasks and assess their behavior. Ethical and societal issues surrounding generative AI are also addressed.
Prerequisite: CS 320

CS 421 Autonomous Software Systems 3-3-0
Cross-level listed with CS 521. This course focuses on the construction and supervision of large-scale software systems using automation-assisted development tools in which natural language specifications guide code generation and modification. Students learn to decompose complex requirements, define precise interface and behavioral constraints, and integrate model-generated code into existing build, test, and deployment pipelines. Emphasis is placed on correctness, security, and execution control in non-trivial codebases including operating-system components, networked services, compilers or interpreters, and distributed backends. The course treats natural language as a formal specification layer, preparing students to design, audit, and maintain complex software systems developed under partial automation.
Prerequisites: CS 321, CS 310

CS 425 Ethical Hacking and Penetration Testing 3-3-0
In this course, students explore the role of the Ethical Hacker by practicing hacking techniques in a controlled, ethical setting to better understand how systems are compromised and defended. The course covers common intrusion and evasion techniques, methods for bypassing intrusion detection systems, and strategies for protecting corporate and government data from cyber-attacks. Students study Internet protocols, traffic analysis, filtering and monitoring techniques, and common attack procedures used to compromise systems. Emphasis is placed on hands-on experience with real tools and methods, as well as on designing effective intrusion detection and defense strategies.
All offensive security activities are conducted exclusively within department-approved, isolated laboratory environments or explicitly authorized targets. Unauthorized testing of external systems is prohibited. Students must agree to course-specific authorization and acceptable-use rules governing security tools and data prior to participating in practical exercises.
Prerequisite: CS 325

CS 426 Digital Forensics and Incident Response 3-3-0
This course covers methods and tools for investigating and responding to cybersecurity incidents. Students learn to collect, preserve, and analyze digital evidence from memory, disk, and network sources, with attention to chain-of-custody and reporting. Topics include log analysis, timeline reconstruction, malware investigation, event correlation, and AI-assisted forensic techniques. Labs and projects simulate real-world incidents and require formal forensic analysis and response planning.
Students must agree to course-specific authorization and acceptable-use rules governing security tools and data prior to participating in practical exercises.
Prerequisite: CS 325

CS 427 Cloud and Virtual Infrastructure Security 3-3-0
Cross-level listed with CS 527. This course examines security challenges in cloud and virtualized environments. Topics include IaaS, PaaS, and SaaS models; virtualization and containerization; Kubernetes security; identity and access management; monitoring; vulnerability assessment; and compliance frameworks such as NIST and ISO 27001. Through hands-on labs, students design, secure, audit, and evaluate cloud-native and multi-tenant infrastructures, applying mitigation strategies to realistic attack scenarios.
Students must agree to course-specific authorization and acceptable-use rules governing security tools and data prior to participating in practical exercises.
Prerequisite: CS 325

CS 428 Cyber-Physical and Industrial Control Security 3-3-0
Cross-level-listed with CS528. This course explores the security of cyber-physical and industrial control systems used in critical infrastructure. Students study OT architectures and protocols, including PLCs, SCADA systems, sensors, and real-time control networks. Topics include threat modeling, attack surfaces, segmentation, monitoring, and incident response in safety-critical environments. Applied work focuses on analyzing simulated control systems and designing security measures that balance cybersecurity with operational safety.
Students must agree to course-specific authorization and acceptable-use rules governing security tools and data prior to participating in practical exercises.
Prerequisite: CS 325

CS 457 Database Software Design 3-3-0
This course covers how one can implement a Database Management system. Major topics are storage management, Query processing, and Transaction management. As a basic assumption, data will not all fit in main memory, so algorithms and data structures appropriate for effective disk storage and quick access must be used. For example, one may use index structures such as B-trees or hash tables. We cover parsing of queries and optimizing of query plans. Finally, we cover durability of transactions using logging, and concurrency control for isolation of transactions. Additional topics in distributed databases are also presented.
Prerequisite: CS 307

CS 462 Image Processing 3-3-0
This course will introduce the area of Image Processing and present classical tools and algorithms in the field including: image perception, image acquisition and display, histogram techniques, image restoration, image enhancement, primitive operations for image analysis, segmentation, image transforms, and pattern and object recognition.
Some examples of industrial applications of image processing and some important developments in image processing research will be also addressed.
Prerequisites: CS 304, MAT 192, PHY 101 (or equivalent)

CS 463 Computer Vision 3-3-0
This course is concerned with the computer acquisition and analysis of image data. Computer vision is the construction of explicit, meaningful descriptions of a physical object from images. Emphasis will be placed on: camera models and calibration, image representation, pattern recognition concepts, filtering and enhancing, segmentation, texture, motion from image sequences, deformable models, matching, stereovision, perceiving 3D from 2D images and tracking with dynamic models. The programming projects assigned in this course will make substantial use of the C and C++ programming languages
Prerequisites: CS 304, MAT 192, PHY 101 (or equivalent)

CS 464 Network Programming 3-3-0
This course presents computer networks at a functional level, with strong emphasis on programming distributed applications over a network. Discussion will be based on open networking and application standards such as the TCP/IP protocol suite and the Portable Operating System Interface (POSIX). Topics normally covered are TCP/ IP architecture and programming, the client-server model, network file systems, streaming, tunnelling. Programming distributed applications (in C or C++) is an integral part of the course.
Prerequisite: CS 216

CS 467 Special Topics in Algorithms 3-3-0
The course builds on the techniques covered in CS 317 to present some specialized algorithms in several areas, including Bioinformatics, Computational Geometry, and Network Flow.
Prerequisite: CS 317 or permission of the instructor

CS 469 Special Topics in Computer Science 3-3-0
The course will present topics of current interest or research directions in Computer Science. The course content is expected to vary from year to year to reflect the current interests of students and faculty. It will be offered by arrangement with the department.
Prerequisite: CS 304

CS 471 Graph Theory 3-3-0
An introduction to the combinatorial, algorithmic and algebraic aspects of graph theory.
Prerequisites: CS 304, MAT 200
Note: See MAT 421. Students may not take this course for credit if they have received credit for MAT 421.

CS 486 Introduction to Quantum Computing 3-3-0
Qubits and quantum states, unitary operations, quantum gates and circuits, measurement, and basic quantum complexity concepts. Core algorithms such as Deutsch–Jozsa, Grover search, and Shor factoring are studied, along with an introduction to quantum error correction and quantum cryptographic protocols. Students gain practical experience through simulation and implementation of quantum circuits and algorithms using standard quantum computing frameworks.
Prerequisites: CS 304, MAT 108

CS 498 Capstone Project 6-0-0
This two-semester capstone course is the culminating project experience for senior undergraduate Computer Science students. Under faculty supervision, students work individually or in small teams to design and complete a substantial independent project that integrates knowledge and skills acquired throughout the program. Projects will be evaluated on clarity of assumptions, robustness under failure, and correctness of reasoning, in addition to functional outcomes. The course culminates in a formal written report and a public presentation or demonstration. This course fulfills the capstone requirement for the Cybersecurity Concentration.
Prerequisite: permission of the department

CS 499F Honours Dissertation 6-0-0
The student is required to complete a theoretical or applied project. The subject is arranged with the student's supervisor during the first four weeks of term. A written dissertation is required as well as two seminar presentation.
Note: This course is open only to final year Computer Science Honour Students in the dissertation stream, and only by permission of the department

Mathematics

Faculty

Madjid Allili,
B.Sc.(Algiers), M.Sc., Ph.D.(Sherbrooke);
Professor

Thomas Brüstle,
B.Sc., (Ludwig-Maximilians), M.Sc.,
Ph.D. (Zurich); Professor,
Maurice-Auslander Research Chair

François Huard,
B.Sc., M.Sc., Ph.D. (Sherbrooke); Professor

Trevor H. Jones,
B.Sc.H. (Acadia), M.Sc. (Dalhousie),
Ph.D. (University of New Brunswick);
Senior Instructor

Scosha Merovitz,
B.Sc.(Bishop's), M.Sc.(Dalhousie);
Special Instructor
Chair of the Department

David Smith,
B.Sc., M.Sc., Ph.D. (Sherbrooke);
Adjunct Professor

N. Brad Willms,
B.Math. M.M., Ph.D. (Waterloo);
Associate Professor

Program Overview

Mathematics is the language of the sciences, a language which allows scientists to quantify, model, understand and predict behaviour in an enormously diverse range of phenomena of interest. Simultaneously, Mathematics is often regarded as an art, as it is the creative study of patterns and of problem solving. Mathematics covers a wide range of disciplines including algebra, analysis, combinatorics and discrete mathematics, and differential equations. In first-year courses, mathematics students are joined by other science students, particularly from Physics and Computer Science. In the advanced courses, classes are very small, and some are given on an individual or tutorial basis.

The highest level of specialization is Honours, and Honours programs prepare students for direct entry into graduate work leading to a Master's or Ph.D. degree. All honours mathematics students have an opportunity to study independently and thus develop their reading and problem solving skills, and there is some chance to pursue special interests. The Majors programs provide students with an excellent general preparation for the career world, while not preventing entrance into graduate school (sometimes after a qualifying year). The Majors programs have sufficient electives to allow students to combine their major with a second major or at least a minor (the least specialized type of program) in another discipline. Students are encouraged to add a minor or major and many do so. Popular choices include computer science, physics, music, English, French, Spanish, drama, and philosophy. The Department of Mathematics offers several specialized, interdisciplinary programs, jointly with other departments, including Hispanic Studies and the School of Education.