Table of Contents

Preface

Welcome to the Department of Electrical Engineering and Computer Science.

This Supplemental Calendar provides detailed information about all the programs and courses we offer. It is part of a spectrum of information that is designed to help you plan your studies. Other written information includes, for example, program checklists published via the Student Hub in the Lassonde School of Engineering.

You are not alone in navigating the requirements of your program. Advice from people, as opposed to written material, is just around the corner. Besides the EECS Undergraduate Program Office (located in LAS 1012M) there are also advisors at the Lassonde Student Welcome and Support Centre (located in 105 Bergeron Centre for Engineering Excellence). Indeed, advising is required before each new fall term, i.e., you will not be able to enrol in courses without it. Our professors also welcome discussion about your aspirations and future career directions, whether in graduate studies or rewarding fields of employment.

Our Undergraduate Program Office (LAS1012M) focusses on course-related advising including enrolment, prerequisites, letters of permission and so forth, as opposed to program requirements.

Our courses are carefully designed to build your knowledge and expertise, and this is reflected by the prerequisite structure. In general, it is not possible to take a course without having taken its prerequisites. Please pay careful attention to this in planning how you meet the requirements of your degree program.

The computing and electrical engineering fields are amongst the most important technological drivers of great change in our society. It is important, therefore, that you not only develop the applied and theoretical skills of a professional, but that you also try to obtain an understanding of the impact of your discipline and its tools on society. Our programs require courses outside the disciplines of Computer Science or Engineering in areas where you will broaden your knowledge of societal issues. In planning your course selection, please ask yourself not only which technical or scientific courses will give you a good degree, but also which courses will make you a good professional. That implies a sound technical background, a broad education, professional ethics and a social conscience.

Departmental Contacts

Electrical Engineering and Computer Science (EECS) Department

1012M Lassonde Building (LAS)
York University
4700 Keele Street
Toronto, Ontario M3J 1P3

http://eecs.lassonde.yorku.ca

Phone/Fax (416) 736-5053 / (416) 736-5872

Office hours: 10:00 am – 4:00 pm
(Fridays during June-August: 10:00 am – 3:00 pm)

Chair: Peter Cribb	Tel. (416) 736-5053 Email: chair@eecs.yorku.ca
Vice Chair (Science): Eric Ruppert	Tel. (416) 736-5053 Email: ug@eecs.yorku.ca
Vice Chair (Engineering): Andrew Eckford	Tel: (416) 736-5334 Email: ug@eecs.yorku.ca
Vice Chair (Graduate): Simone Pisana	Tel. (416) 736-5053 Email: gpd@eecs.yorku.ca

Faculty

	Telephone Extension	email @eecs.yorku.ca		Telephone Extension	email @eecs.yorku.ca
Aboelaze, Mokhtar	40607	aboelaze	Lian, Yong (Peter)	44647	peterlian
Allison, Robert	20192	allison	Litoiu, Marin	20987	mlitoiu@ yorku.ca
Amanatides, John	44782	amana	Ma, Burton	77885	burton
An, Aijun	44298	aan	Mackenzie, Scott	40631	mack
Baljko, Melanie	33348	mb	Magierowski, Sebastian *	44652	magiero
Brown, Michael	66675	mbrown	Mirzaian, Andy	70133	andy
Brubaker, Marcus **	77883	mab	Nguyen Uyen	33274	utn
Cheung, Gene	77884	gcheung	Ostroff, Jonathan	77882	jonathan
Chinaei, Amir	33947	ahchinaei	Papangelis, Manos	44782	papaggel
Cribb, Peter	44149	peterc	Pisana, Simone	77885	pisana
Datta, Suprakash	77875	datta	Rezaei Zare, Afshin	70152	rezaei
Dymond, Patrick *	33948	dymond	Roumani, Hamzeh	66146	roumani
Eckford, Andrew	33928	aeckford	Ruppert, Eric	33979	ruppert
Edmonds, Jeff	33295	jeff	Sadeghi-Naini, Ali	20590	asn
Elder, James	66475	jelder	Smith, James A. *	33978	drsmith
Faloutsos, Petros	40630	pfal	Sodagar, Amir	70117	sodagar
Farag, Hany	33844	hefarag	Spetsakis, Minas	77886	minas
Ghafar-Zadeh, Ebrahim	44646	egz	Stachniak, Zbigniew	77877	zbigniew
Godfrey, Parke	66671	godfrey	Tabassum, Hina	20889	hina
Grau, Gerd	70127	grau	Toptsis, Anestis **	66675	anestis
Gryz, Jarek	70150	jarek	Tourlakis, George	66674	gt
Hooshyar, Ali **	33939	hooshyar	Tsotsos, John	70135	tsotsos
Hornsey, Richard	33265	hornsey	Tzerpos, Vassilios	33341	bil
Jenkin, Michael	33162	jenkin	Urner, Ruth	70128	ruth
Jiang, Hui	33346	hj	van Breugel, Franck	77880	franck
Jiang, Jack	33939	zmjiang	Vlajic, Natalija	77878	vlajic
Kassiri, Hossein	77874	hossein	Wang, Jackie	70130	jackie
Kyan, Matthew	33965	mkyan	Wang, Ping	70125	pingwang
Lam, John	77872	johnlam	Wildes, Richard *	40203	wildes
Lesperance, Yves	70146	lesperan	Xu, Jia *	77879	jxu

* On Sabbatical
** On Leave

Programs Offered by the Department

The Department offers courses towards the following programs, each of which is described more fully below.

1. Computer Science
2. Computer Security
3. Digital Media
4. Computer Engineering
5. Software Engineering
6. Electrical Engineering

In this document, the BA or BSc degree refers to the 90-credit bachelor degree. The BA Honours or BSc Honours degree refers to the 120-credit degree. The BEng is a specialised honours, typically 150-credit, engineering degree.

For detailed information you are advised to first read the appropriate sections of the York University Undergraduate Calendar (http://calendars.registrar.yorku.ca). Secondly, read this supplemental Calendar, and thirdly, see an advisor in the department or in the Bergeron Student Welcome and Support Centre.

Current degree requirements for the programs offered by the Department are listed at http://lassonde.yorku.ca/program-checklists. Checklists for previous years are available via the departmental web page.

Computer Science

Computer Science is available as a major program leading to an Honours or a Specialised Honours (120-credit) degree. It is also available as a 90-credit Bachelor degree. Students in an Honours or Specialised Honours degree program may also graduate with the 90-credit degree once they have fulfilled its requirements, and then continue to obtain their Honours degree. The degree types are: BA Honours, BSc Honours, BA or BSc Specialised Honours, International BSc (iBSc) Honours and International BA (iBA) Honours, and the International Dual Degree (BSc Specialised Honours/York; BSc Bachelor/Hochschule Bonn-Rhein-Sieg).

The Honours major in Computer Science may be combined with many subjects in the Faculties of Science, Environmental Studies, Health, or Liberal Arts and Professional Studies (LA&PS), the Lassonde School of Engineering or the School of Arts, Media, Performance and Design. Such a combination can lead to a four-year double major or major-minor degree. Conversely, Computer Science is also available as a Minor program, which must be combined with an Honours Major in a different discipline.

The intention of a combined program is for students to major in two subjects. In a double major program, students complete course work up to and including the 4000-level in each subject. In a major/minor program the minor subject generally requires somewhat less course work than the major, and still may include courses at the 4000-level. Such combined degrees may require students to take more than the minimum of 120-credits in order to satisfy the honours requirements of each subject. Consult advisors in both departments if you are planning a combined program.

In the Specialised Honours program students take more courses in computer science and mathematics than in other programs, thereby achieving a greater depth of study. However, a breadth of education is maintained by the requirement of a significant number (30 credits) of courses outside of computing, math and statistics.

The BA Honours and BSc Honours programs require 120 credits (normally completed in four years of study), more specialization, a higher minimum performance level (grade-point-average of 5.00 to proceed[1] — i.e., continue in the program — and to graduate), and in some cases different courses than a BA or BSc degree.

The 90-credit BA and BSc programs, normally completed in three years of study, require a minimum grade point average of 4.00 over all courses for graduation.

The required courses in computer science and mathematics are identical in most computer science programs in the first two years of study so that students can make their final decision as to which program to graduate in after they have more exposure to the discipline. All computer science programs are structured in such a way that a student who embarks on a

[1] *Students may "**proceed on warning**" if they fail to meet the gpa of 5.0. The **minimum** cumulative gpa **required** is 4.00 between 0-23 credits; 4.25 between 24-53 credits; 4.80 between 54-83 credits; 5.00 beyond 83 credits.*

BA Honours or BSc Honours program can meet the requirements for a BA or BSc Bachelor degree (90 credits) by the end of the third year, and can at that time graduate with either a BA or BSc Bachelor degree. Only the honours programs (excluding the minor) are accredited by the CSAC.

Software Development Stream
The Specialised Honours programs (BA and BSc) may be taken with a specified focus (specialization) or *Stream in Software Development*. The stream provides a mechanism for recognising on your transcript this emphasis or focus in your studies. It *requires* some specific 3000- and 4000–level courses (thus specifying what would otherwise be free choices within EECS courses that you would make yourself in an un-streamed Specialised Honours program), as well as a full year (6-credit) 4000–level project, or “honours thesis” as it would be called in some universities. This is EECS 4090 6.00.

iBSc and iBA
The department has a strong interest and involvement in promoting opportunities for students to study abroad. The iBSc and iBA Degree programs are structured as honours computer science programs that contain a compulsory exchange placement abroad of at least one full term of study. The iBSc Degree program requires 30 credits outside the major, consisting of 12 to 18 credits in a language chosen by the student, and another 12 to 18 credits that focus on a country or region that is compatible with the student’s chosen language and/or consistent with an international issue that is of interest to the student. The iBA also requires 30 credits like the iBSc above, but the language component is set to exactly 18 credits in this degree. Students would normally enrol in language courses relevant to their exchange placement.

International Dual Degree BSc Specialised Honours
In collaboration with the Departments of Computer Science in the Hochschule Bonn-Rhein-Sieg (BRSU) and the University of Crete (UoC), the Department of Electrical Engineering and Computer Science offers an *International Dual Degree Program in Computer Science* (BSc and Specialised Honours BSc).

- It is an international program of study at York University, BRSU and UoC that equips the graduate with professional credentials in North America and in Europe.
- Two degrees are obtained within four years of study: The York University BSc Specialised Honours Degree in Computer Science and the BRSU Bachelor of Science Degree in Computer Science.
- The program includes a 1-year long study in Europe, divided between BRSU and UoC.

This program will be of interest to students with high academic standing as measured by a cumulative GPA of 6.00 or higher, computed over *all* major computer science (EECS) courses completed among approximately 60 credits taken at York University (typically achieved at the end of the second year of study).

Students in the program, after two years of study at York but before the completion of the York degree requirements, will be eligible to continue their studies as York international exchange students in the European Union *for a full year of study*. This exchange placement will be divided between BRSU and UoC. At UoC, the students will complete a mandatory *research internship* component and an *undergraduate thesis.* Students may also take computer science courses at UoC.

The thesis and internship activities will be conducted in English.

The program of study is precisely regulated to meet both the degree requirements of York University and BRSU. At the end of year three (the exchange year), students who have progressed normally will have met both the BRSU Bachelor of Science in *Informatik* (equivalent to York's BSc Bachelor 90-credit degree) requirements as well as those of the BSc Bachelor in Computer Science (York), and may graduate with both of these degrees from the respective institutions. York students will return and complete a fourth year of study at York University to fulfil their BSc Specialised Honours degree requirements and thus also graduate with the York University Specialised Honours degree. York International administers all exchanges under this program in collaboration with the International Offices in BRSU and UoC.

Reciprocally, BRSU students spend a full year of study at York to conclude their third-year BRSU requirements by taking York University degree-specific substitute courses. Upon successful completion of year three, these students would have met both the BRSU and York degree requirements, and would be eligible to earn the York BSc Bachelor degree in Computer Science (as well as the Bachelor of Science in Informatik from their home university).

All BRSU students in this Dual Degree Program must satisfy a *modified general education requirement*, as follows: They must complete at BRSU, normally prior to arrival at York, 18 ECTS (the equivalent of 9 York credits) of courses in English, Microeconomics, Intercultural Communications, and Law.

Reciprocally, all York students in this Dual Degree Program must satisfy a *modified general education requirement*, as follows: They must complete at York University 6 further non-science credits in addition to 12 credits in language and culture courses.

Computer Security
The Computer Security program is a Specialised Honours degree that may be pursued as a BSc or a BA degree program. It focuses on understanding threats to computer security and the techniques for combating those threats. Besides the foundational computer science and mathematics courses the program requires in-depth education in areas such as computer networks, cryptography, operating systems, databases, and software engineering techniques as well as specialised courses in computer security. In addition, a solid understanding of applied ethics, management and operational practices, and exposure to relevant legal concepts are important elements of the curriculum.

As a specialised honours program, computer security cannot be combined with any other honours major or honours minor. However, the program does still require a significant number of non-EECS and non-MATH courses to ensure a breadth of general education.

Digital Media
Digital media or new media are the technical methods and social practices of communication, representation, and expression that have developed using the digital, multimedia, and networked computer. Digital media have

transformed work in other media (books, movies, telephones, television) and given rise to entirely new media (computer games and the internet for example).

The curriculum aims to provide a foundation in the following areas:

- The computational basis for the creation of digital media imagery and sound, including animation and the simulation of 3D environments.
- The theoretical, artistic, aesthetic and experiential ideas that lie behind an informed understanding of the aesthetic aspects of digital media creation
- The practice of creating digital media works that explore the ways in which culture is produced and can be produced through technology
- The broader socio-cultural effects and the theory and research concerning responses to and uses of digital media.

This is a multidisciplinary BA Specialised Honours degree program that consists of a common core of courses and is structured as three distinct specialisation *Streams*: *Digital Media Development, Digital Media Arts* and *Digital Media Game Arts*. A student must choose one of the streams by the end of the first year of study.

A 90-credit Bachelor BA is also available as of fall 2016, but not as a direct entry option. All streams involve a nearly balanced number of courses from the Department of Electrical Engineering and Computer Science (EECS) and the School of the Arts, Media, Performance and Design (AMPD). There are also a few courses required from Science and Technology Studies (STS) in the Faculty of Science (FS) and a few from the Communication Studies Program in the Faculty of Liberal Arts and Professional Studies (LA & PS).

Computer Engineering

This is a Specialised Honours Bachelor of Engineering (BEng) Degree Program in which students must select courses that focus on software and hardware engineering. For example, courses in digital logic, embedded systems, signals and systems, and computer networks are required in Computer Engineering but are optional for students in other degree programs. Moreover, the BEng degree contains a substantial core of engineering design courses that are only open to students in an Engineering program.

While Honours programs in Computer Science allow flexibility for students to choose electives, the Computer Engineering program is highly specified in order to meet accreditation requirements of the CEAB. Computer Engineering is our oldest BEng program and is already accredited by CEAB.

As is the case with all engineering programs, the workload is very demanding. The total number of credits (normally completed over four years of study) is 150.

Software Engineering
This Specialised Honours BEng Degree Program applies computer science and engineering principles to the creation, operation, and maintenance of software systems including embedded systems (e.g. devices such as mobile phones or air traffic systems controlled by software) ubiquitous in modern technology. Skills in Software Engineering are increasingly in demand given the prevalence of software and its use in critical areas involving the safety of the public and environment.

York’s Software Engineering program provides students with a systematic and disciplined approach to developing mission critical software. The software engineering curriculum at York University develops the multidisciplinary skills required by today's software engineers—technical, mathematical, business, societal, and communication—that really make software engineers the leaders of tomorrow. The program develops teamwork, communication skills (via technical presentations, reports, and peer evaluations) and encourages an industrial internship.

The first year provides students with a strong foundation in programming, applied mathematics, and physical sciences.
During the second and third years, Software Engineering students acquire the necessary engineering tools in mathematics, computer and engineering sciences, as well as specialised skills in software specification, for the analysis and design of complex mission critical systems by combining intensive classroom teaching and laboratory education.

The fourth year of the Software Engineering curriculum is flexible to enable students to create their own specializations by selecting from a variety of innovative courses in the fields of net-centric computing, mobile communications, security, databases and human-computer interfaces. Multidisciplinary skills in social sciences, business, humanities, and communications are honed through a selection of elective courses in complementary studies spread throughout the four years of the curriculum.

Design is a significant component of engineering and is integrated throughout the software engineering curriculum. In addition to the standard engineering design courses, the design of software is stressed throughout. There is a software project course in the second year as well as a design project in the third year. The design process culminates with a capstone engineering project in which students put their training into practice by developing requirements, designing a suitable architecture, building, testing and deploying a software intensive system ideally in an interdisciplinary environment.

Electrical Engineering
Electrical Engineering deals with the electrical, electronic, and wireless infrastructure that enables our modern life. There isn't a field or industry that doesn't depend on the fundamentals of Electrical Engineering – be it pharmaceutical, medical, manufacturing, media or even entertainment. Sub disciplines of Electrical Engineering include electronics and nanoelectronics, control systems, telecommunications, robotics systems and biomedical instrument design. Common across all engineering programs, the first year in Electrical Engineering provides students with a strong foundation in programming, applied mathematics, and physical sciences. During the second and third years, the curriculum covers advanced topics in electronics and electrical circuits, semiconductor devices and circuits, electromagnetic fields and waves, power systems and energy conversion. The final year provides students the flexibility of specializing in one of the four electrical engineering fields: Electronics, Power, Communications and Signal Processing, and Medical and Assistive Devices.

Engineering and International Development Studies
Computer, Electrical or Software Engineering students can combine their engineering studies with specialization in international development as a dual degree. Students in the engineering and international development studies program choose one of the engineering program streams, plus the international development studies requirements. Graduates are awarded

both BEng and BA degrees. See the university calendar for details http://calendars.registrar.yorku.ca

CSAC and CEAB Accreditation

EECS offers the following accredited programs:

- BSc Specialized Honours in Computer Science. Accredited by CSAC since 1996.
- BA Specialized Honours in Computer Science. Accredited by CSAC since 1996.
- BSc Honours Major (including Double Major). Accredited by CSAC since 1996.
- BA Honours Major (including Double Major). Accredited by CSAC since 1996.
- BA Honours in Computer Security. Accredited by CSAC since 1996.
- BSc Specialized Honours in Computer Security program. Accredited by CSAC since 1996.
- BEng Computer Engineering. Accredited by CEAB since 2007.
- BEng Software Engineering. Accredited by CEAB since 2016.
- BEng Electrical Engineering. Accredited by CEAB since 2017.

CSAC is an autonomous body established by the Canadian Information Processing Society (CIPS), while the CEAB was established by Engineers Canada. The purpose of accreditation is to identify those institutions that offer programs worthy of recognition. The objectives of the accrediting bodies are:

▪ To formulate and maintain high educational standards for Canadian universities offering computer, engineering and information science programs, and to assist those institutions in planning and carrying out education programs.

▪ To promote and advance all phases of computer, engineering and information science education with the aim of promoting public welfare through the development of better educated computer professionals.

- To foster a cooperative approach to computer, engineering and information science education between industry, government, and educators to meet the changing needs of society.

Graduation from an accredited Computer Science Program simplifies the process of professional certification as an Information Systems Professional of Canada or ISP. The provinces of Ontario and Alberta recognise the ISP designation. Likewise, accreditation from CEAB ensures that the academic requirements necessary for registration as a professional engineer within Canada are successfully met. More information on professional accreditation and the accreditation process can be found on the CIPS web page at http://www.cips.ca/accreditation and on the Engineers Canada website at http://engineerscanada.ca/accreditation/about-accreditation

Co-op and Internship Options

The department offers a range of opportunities to students to augment their studies with paid work experience. For more information, see http://www.lassondecoop.com.

The Internship and Co-op options are administered centrally by the Lassonde School of Engineering. Internship students receive assistance in identifying relevant and interesting internship opportunities, formulating the employer application package and sharpening their interview skills. Students have been placed at a wide range of companies including IBM, Blackberry, Sun Microsystems, Platform, Workbrain, Ontario Lottery and Gaming Commission, CIBC, Toronto Hydro, Ontario Power Generation, Global Matrix, Shore Consulting Group, RBC, and the Ontario Ministry of Government & Consumer Services.

Co-operative Education

Students in our engineering, computer science or computer security programs have the opportunity to participate in Co-op Work Terms, typically starting in the summer after second year. (This option is available to computer science and security students who start in 2018 or later.) There is considerable flexibility in the scheduling and duration of individual co-op internships but a minimum of 12 months of co-op work is required. The type of internship is also flexible with traditional, entrepreneurial, international and freelance placements possible. During the co-op internship placement students earn a salary typical of entry-level positions in the field.

Internship

The *Professional Experience Program* offers qualified undergraduate Computer Science, Computer Security, and Digital Media students the opportunity to take part in a single work placement that lasts 4, 8, 12 or 16 months. These are paid internships and, for example, computer science/security interns typically earn a salary typical of entry-level positions in the IT sector.

Students in the BA Honours, BSc Honours, iBSc, iBA are eligible to apply to the PEP in year three of their studies. Students enrolled in the Internship option are required to enrol in EECS3900 0.00 in *each term* of their internship. For administrative reasons we have separate courses, EECS3980 0.00, and DATT 3929 0.00 associated with Internships of Computer Security and Digital Media majors.

Industrial Partnership

The industrial partnership option allows computer science students to work part-time throughout their four-year degree with our industry partner, Shopify. Work experience and on-the-job training in this program will provide up to 24 credits towards a B.Sc. Honours degree in computer science.

Opportunities to Study Abroad

In addition to the international B.Sc. and B.A., the international dual degree (which are described above), there are other opportunities for York students to study abroad.

Since 2003 the Department has maintained a successful International Summer School program, mounting summer courses in partnership with departments in Germany, Greece and Poland. This program allows students to take York computer science courses at universities in Europe.

For more information on this and other international study opportunities, see the following web page:
http://eecs.lassonde.yorku.ca/activities/international-opportunities/

Degree Requirements

Specific course requirements for the degree programs outlined above can be found in the official University Calendar at:

http://calendars.registrar.yorku.ca

Degree requirements fall into two or three broad categories:

1. Those required for the major, i.e., EECS and mathematics courses; and for the Digital Media degree, courses from the AMPD School and social science courses/STS courses.
2. Those required for the second major, or minor, if the program is an Honours Double Major or Honours Major/Minor program.
3. Courses required for General Education, breadth and diversity. These depend on whether the degree is a BA, BSc or a BEng.

The School provides degree checklists that itemize the course requirements. Every effort is made to ensure the accuracy of these checklists, however, in case of any inconsistency the official university calendar is to be followed. These checklists are available from

http://lassonde.yorku.ca/program-checklists.

Courses on Offer in 2018-19

The course schedule for Summer 2018 and FW 2018-19 is on the department Undergraduate page for continuing students

http://eecs.lassonde.yorku.ca/current-students/undergrads-courses/

under the heading "**Course Information—Lecture Schedules**".

Admission to Programs

Computer Science and Computer Security Programs

Please go to http://futurestudents.yorku.ca/requirements/ to find out about the various University and Faculty level Admissions Requirements pertaining to your situation. There are two general Admission Categories:

1. Entry with only secondary school background

Requirements under this category are detailed at
http://futurestudents.yorku.ca/requirements/highschool
Please note the Faculty-specific requirements as these pertain to your case.

2. Entry with post-secondary academic background
Please follow http://futurestudents.yorku.ca/requirements/univ_coll to find a detailed description of general University and Faculty-specific policies for gaining admission under this category.

In particular, current York University students who want to change their major to be, or include, Computer Science or Computer Security will need to meet the following *minimum* requirements:

- Completion of at least 24 credits with an overall cumulative grade point average (OCGPA) of 5.00 (grade of C+) or better if transferring to the honours computer science/computer security programs (minimum OCGPA of 4.00 (grade of C) is needed to transfer into the Bachelor degree programs in Computer Science).
- *Must* meet *either* the mathematics component of the published Admissions to the Degree Program requirements completed within the last 5 years, *or* the prerequisite alternatives (2) or (3) in the prerequisite of EECS1012. Qualifications for entry cannot be mixed (entry is decided either on Admission qualifications alone, or on *one* of the two alternatives above). *Once courses from alternative (2) are taken, a transfer application cannot be based on criteria (1) or (3) anymore.*

Once transferred to a EECS Program, students will need to satisfy all specific and general prerequisites of computer science courses they wish to take.

Digital Media Program
Admission requirements can be found at the same websites as given above. Please go to http://futurestudents.yorku.ca/requirements/.

Electrical, Computer and Software Engineering Programs
Admission requirements can be found at the same websites as given above. Please go to http://futurestudents.yorku.ca/requirements/.

Graduate Programs in EECS

York University has the only graduate program in Electrical Engineering & Computer Science in Canada. It provides you with breadth, as well as specializations in Computer Engineering, Computer Science, Electrical Engineering and Software Engineering. Our program has 140 students supervised by 60 professors, whose research ranges from biomedical engineering to virtual reality and from data mining to nanoelectronics. More detailed information about specific research topics can be found on the websites of the research groups and professors associated with our program. Graduates of our program have ended up both in academia and industry. Former students can now be found at companies such as Amazon, Apple, Facebook, Google, IBM, Microsoft, NASA and Uber. In our program, you can pursue a Master of Science (Computer Science), a Master of Applied Science (Computer Engineering, Electrical Engineering, or Software Engineering), or a Doctor of Philosophy (all four fields of specialization). For more information, please visit:

http://eecs.lassonde.yorku.ca/programs/graduate

Exceptional undergraduate students may apply for permission to enrol in a graduate-level EECS course by contacting the EECS office. This requires permission of both the undergraduate programme director and the graduate programme director. Such a course could normally be used as a substitute for a 4000-level EECS course in the undergraduate degree requirements.

Out of Major Elective Courses - Computer Science and Computer Security Programs

Students in Computer Science or Computer Security sometimes feel their study in this discipline is quite isolated from the other programs in their Faculty, and place little emphasis on their choice of courses outside the major, even though at least a quarter of their courses are non-computer science/math. This is a mistake — computer science supports applications in every information-using discipline. In order to make creative and effective use of your skills in computing, you need to know much more of the natural world, the man-made world, and the world of ideas, than can be learned in courses in computing alone.

There are many choices for elective courses beyond computing. For example courses in economics, philosophy (logic), psychology, linguistics, physics and chemistry, just to name a few, whose content meshes with issues and problems studied in computer science.

Not only should you consider taking individual courses in other disciplines but you should also consider taking a concentration of non-major courses that together form a coherent or complementary package. Such a concentration may come from only one discipline but it may also come from two or three disciplines on related concepts presented from different perspectives. It will often be necessary to take specific prerequisites before you can take a desired elective course; such combinations also form coherent concentrations.

To further emphasise the importance of elective courses outside the discipline, all honours programs require at least 30 credits from non-"IT" and non-"MATH" courses.

EECS Courses for Non-Majors

The Department also offers a variety of courses at the 1000-level and 2000-level that are of interest to students wanting to learn about computers and computer use without majoring in Computer Science or Engineering. In some cases degree programs offered by other departments may require these courses in their programs.

At the 1000-level these courses for non-majors are:

EECS1520 3.00 Computer Use: Fundamentals
EECS1530 3.00 Computer Use: Programming
EECS1540 3.00 Computer Use for the Natural Sciences
EECS1541 3.00 Introduction to Computing for the Physical Sciences
EECS1550 3.00 Introduction to Web Development
EECS1560 3.00 Introduction to Computing for Math and Statistics
EECS1570 3.00 Introduction to Computing for Psychology

EECS1520 is an introduction to computers including their architecture, system software, networking and other general topics as well as providing exposure to problem solving applications such as the spreadsheet. The course EECS1530 is an introduction to computer programming and may be taken as preparation for EECS2501. EECS1550 is an introduction to the development of interactive web applications. EECS1541, EECS1560 and EECS1570 are directed towards Physics and Astronomy, MATH/Stats and Psychology majors, respectively.

These courses are not appropriate prerequisites for most EECS major courses. Students interested in taking further EECS courses should begin with EECS 1011, EECS 1012 or EECS1710. Some students might find EECS 1530 useful preparation before attempting EECS 1011, EECS 1012 or EECS1710.

At the 2000-level the Department offers the course EECS2501 1.00, Fortran and Scientific Computing, which covers computer-based problem solving in a variety of scientific and engineering settings.

Extra-Curricular Activities

The clubs and student organizations at Lassonde offer some fantastic opportunities and plenty of fun alongside personal development. Please visit the Lassonde Student Clubs website (http://lassonde.yorku.ca/ clubs) for the complete list our clubs and student organizations.

The **Computing Students Hub** (CSHub) is a group of students passionate in computer science and technology. While CSHub represents computing students in Computer Science, Computer Security, Digital Media, Computer Engineering and Software Engineering, the club has members from all Faculties. CSHub offers seminars/tutorials/workshops on a variety of topics, programming competitions, lectures from distinguished individuals in industry, as well as social and professional development events. Please visit their website (http://www.cshub.ca) for more details about the club and its events.

The **Digital Media Student Association** (DMSA) is a group of students who offer academic, social and career support to students within and beyond York University’s Digital Media Program. More information about the DMSA is available on their website at http://dmstudents.ca/.

The **Lassonde Engineering Society at York**, or EngSoc, represents Engineering students on various issues relating to engineering and the university, and organises social events and advising sessions. They can be accessed through their website at http://engsocyu.com/.

The **York Programming Contests** are held throughout the year. A York Programming Champion cup is awarded each year, and a team of students is chosen to represent York at the regional ACM International Collegiate Programming Contest. See http://wiki.eecs.yorku.ca/project/ACM for more information.

The Student Ombuds Service

The Student Ombuds Service (SOS) is a peer-advising service designed to help York students find university-related information that they need. The SOS office is staffed with knowledgeable upper-level students and serves as a resource centre and the hub of a referral network, assisting students to find answers to any questions about York University policies and procedures, giving general academic help, and advice about University life. SOS resources include departmental mini-calendars, graduate and professional school information, a tutor registry, and a study group registry. The SOS office is located in 208 Bethune College and has drop-in hours between 10:00 a.m. and 4:00 p.m., Monday to Friday. No appointment is necessary. SOS can also be reached on the web: http://www.yorku.ca/sos.

Teaching Laboratories

Undergraduate students who are registered in EECS courses use the Department of Electrical Engineering and Computer Science undergraduate computing laboratories. The majority of students are granted an authorised account through which they can store, print or electronically submit files related to their course work, and create personal web sites. Students access the Linux or Windows workstations in the laboratories through scheduled sessions or on a drop-in basis. The accounts can also be accessed remotely through the internet via a secure connection, or from other designated laboratories on campus. Select laboratories are equipped with printing facilities. First-year students use a dedicated Computing Laboratory to learn about concepts of computing within a heavily equipped experimentation environment. Senior students use a variety of specialty laboratories. These include the Generalized

Signal Processing, the Power Engineering, the Electronics, the Electrical Characterization and Testing, the Medical Devices, the Software Engineering, the High Performance Computing, the Computer Security, and the Robotics and Virtual Reality Laboratories.

- The Generalized Signal Processing Laboratory incorporates Robotics, Graphics and Virtual Reality Laboratories. It consists of two CRS robot arms, an autonomous mobile robot, workstations equipped with multimedia hardware including monocular and stereo video cameras and audio facilities. The Virtual Reality Laboratory supports the study of modern virtual reality systems with a variety of specialised hardware displays and tracking devices, a large-screen passive stereoscopic display, Phantom Omni haptic devices, immersive audio displays, and a number of magnetic and inertial motion tracking devices. The Generalized Signal Processing Laboratory also incorporates The Digital and Embedded Systems Laboratory to provide hands-on experience in digital logic design, connecting discrete components such as gates, flip-flops and registers on integrated circuit chips. Students are also exposed to design on FPGA boards using hardware description languages. It consists of embedded microcontroller boards, logic analysers, oscilloscopes and other electronic test equipment to provide students with hands-on experience with the design and implementation of digital and embedded systems.
- The Software Engineering Laboratory consists of a project meeting area and a work area with Linux and Windows workstations equipped with modern software development tools to provide students experience with various phases of the software development life cycle such as requirements, analysis and design, implementation, testing, delivery, and maintenance.
- The Electronics Teaching Laboratory contains tools and electronic measurement equipment for physical systems electronics applications (computing and digital communications systems and applications such as low-level component design for analog circuits, microwave circuits/devices, semiconductor devices/sensors, electromagnetic) to support the electrical engineering curriculum and provide students an opportunity to gain experience with more advanced equipment not only in a standard class-based setting, but as part of their own project-based

learning experience. The lab serves as a general educational laboratory space for a large number of courses in physical and systems electronics. It provides basic equipment for labs and projects which, for upper-year courses, are augmented by the broader array of test and characterization hardware available in the Analog Support Laboratory, which includes semiconductor devices and sensors, integrated circuits, signal conditioners and instrumentation, electromagnetics and antennas, electro-optics, analog communications and remote sensing and is outfitted to support a broad range of operating signals (DC to 67 GHz, nV to 60 V, pA to 10 A) and test-methods (time-based, frequency-based, network parameters, linearity, noise, parametric) and the Digital Support Laboratory which provides our upper-year students access to high-tech devices for course or capstone projects or even their own entrepreneurial pursuits. The courses supported by DSL are very broad and include digital logic design, embedded electronics, computer architectures, very-large-scale-integration, internet-of-things, digital and wireless communications.

- The Electrical Characterization and Testing Laboratory contains a wide variety of equipment, such as noise figure or semiconductor parameter analyzers, PNA-X microwave network analyzer, logic analyzer, semi-automatic probe stations, arbitrary waveform generator, microwave analog signal generator, pulse pattern generator, vector signal generator, wide bandwidth scope, power supplies, die bonder, source meter, shielding measurement box, and soldering stations.
- The Medical Devices and Integrated Signal Processing Laboratory consist of a number of advanced data acquisition and analysis life science research platforms to provide signals and physiological measurements and analysis of ECG, EEG, EGG, EMG, EOG, etc. Inverted microscopes, MRI and Ultrasound devices, electrochemical and electrophysiological devices for the support of medical devices courses complement the laboratory. This is an advanced undergraduate laboratory offering the students the opportunity to design, implement and test medical devices and biological instrumentation. The support laboratory is dedicated (1) to the preparation of cellular and molecular biological samples and (2) to the testing and characterization of biological samples micro-fluidic, BioMEMS devices. This laboratory is

dedicated to culture the neuronal cells or small animals using a variety of equipment including the incubators, refrigerators, etc.

- The Power Systems Teaching Laboratory (PSTL) is an educational laboratory space for a large number of upper-year courses in power engineering and energy systems. This laboratory allows students to develop hands-on skills and practical experience in various aspects related to power systems engineering. Students will have the opportunity to use power measurement tools and power system workstations to design, implement and test different types of power circuits, electric machines, and to study the behaviour of distributed power networks for renewable energy applications.
- The Computer Security Laboratory consists of virtualized Windows and Linux platforms with specialised software tools and hardware equipment for computer security courses. This is an Internet-accessible, IP traffic-isolated, virtual lab that allows students to experiment with network configuration, security vulnerabilities, and malware without the risk of infecting the campus network.
- The Eshrat Arjomandi Laboratory consists of Linux-based workstations equipped with software such as various compilers and other development environments for courses at the 2000 level and up (e.g., Java, C, Python, Eclipse, Eiffel, Verilog, SPIM). The laboratory is used for classroom instruction, tutorials, student work, and in-lab tests.

All computers in the Department are connected to the campus network backbone, providing access to all significant systems and services in the University, as well as the internet. All laboratories have access points to provide wireless internet access.

Computer Use Policy

Working in a laboratory environment requires cooperative behaviour that does not harm other students by making any part of the Department's computer systems unusable such as locking out terminals, running processes that require lots of network traffic (such as playing games on multiple terminals), or using the facilities to work on tasks that are not related to course work. Essentially, all users of common facilities need to ask themselves whether or not their behaviour adversely affects other users of the facility and to refrain from engaging in "adverse behaviour".

Good manners, moderation and consideration for others are expected from all users. Adverse behaviour includes such things as excessive noise, occupying more space than appropriate, harassment of others, creating a hostile environment and the displaying of graphics of questionable taste. Lab monitors are authorised to ensure that no discomfort is caused by such practices to any user.

The Department policy on computer use prohibits attempting to break into someone else's account, causing damage by invading the system or abusing equipment, using electronic mail or file transfer for abusive or offensive materials, or otherwise violating system security or usage guidelines. As well, we expect you to follow Senate policies (please follow the link on the related Senate Policy

http://www.yorku.ca/secretariat/policies/document.php?document=77)

The Academic Systems Coordinator, in conjunction with the Department and York Computing Services, will investigate any suspected violation of these guidelines and will decide on appropriate penalties. Users identified as violating these guidelines may have to make monetary restitution and may have their computing privileges suspended indefinitely. This could result in your being unable to complete courses, and a change in your major.

Adverse behaviour may also violate University, Provincial and Federal laws; for example duplication of copyrighted material and theft of computer services are both criminal offences. In such cases the University, Provincial or Federal authorities may act independently of the Department. The police may be asked to investigate and perpetrators may be liable for civil and/or criminal prosecution. The Department does not assume any liability for damages caused by such activities.

Awards

Unless otherwise stipulated, students in the Lassonde School of Engineering are eligible for these awards. The Department maintains plaques commemorating the achievement awards.

Computer Science Academic Achievement Awards

Up to four cash awards are presented annually, one for each of the four years of study, to Honours degree students who are majoring in any of the programs offered by the Department and achieved the highest cumulative standing. These awards are funded by contributions from the Department and are the following:

- Marvin Mandelbaum Academic Achievement Medal: awarded annually in recognition of outstanding academic achievement in 1st year and enrolled in an Honours Degree program majoring in any of the programs offered by the Department of Electrical Engineering and Computer Science.
- Michael McNamee Academic Achievement Medal: awarded annually in recognition of outstanding academic achievement in 2nd year and enrolled in an Honours Degree program majoring in any of the programs offered by the Department of Electrical Engineering and Computer Science.
- Anthony Wallis Academic Achievement Medal: awarded annually in recognition of outstanding academic achievement in 3rd year and enrolled in an Honours Degree program majoring in any of the programs offered by the Department of Electrical Engineering and Computer Science.
- James Mason Academic Achievement Medal: awarded annually in recognition of outstanding academic achievement in 4th year and enrolled in an Honours Degree program majoring in any of the programs offered by the Department of Electrical Engineering and Computer Science.

Other Awards

- Students in the Department are encouraged to apply for Summer awards such as the NSERC Undergraduate Summer Research Award. These awards pay students a salary over the summer while they are working on a research project under the supervision of a faculty member. Normally students who have completed at least their 2nd year may apply and typically a grade point average of at least 7.00 (B+) is required. In addition, faculty members sometimes employ undergraduate research assistants over the summer period. Such positions are only offered to the students in the Department with the highest academic standing and demonstrated promise in research and to a very small number of external applicants of similar qualifications.

For more information refer to: http://www.lassondeundergraduateresearch.com/

- There are many additional awards, bursaries and scholarships on offer to fund your studies at the Lassonde School of Engineering. For further details on these opportunities, please consult the following website: http://lassonde.yorku.ca/awards-bursaries-scholarships.

Awards Administered by Student Financial Services

The awards listed below highlight a sampling of those available to students in the Lassonde School of Engineering. Students are encouraged to consult the Student Financial Services' website (http://sfs.yorku.ca/scholarships/) for a comprehensive list of available scholarships, bursaries and awards. Specific details regarding eligibility, criteria and the application process are also available on the website.

- CGI Award – available to undergraduate students majoring in computer science or information technology who have a minimum cumulative grade point average of 6.00 (B).
- Charma Mordido Figuracion Bursary – awarded annually to a female computer science major.
- GM Bursary for Undergraduate Students in Computer Science – available to undergraduate students in computer science, administered by Student Financial Services.
- Hany Salama Bursary – a cross-Faculty bursary available to ITEC, MATH and EECS students who have completed a minimum of 30 credits.
- Mary Stevens Memorial Bursary – a bursary awarded to a mature student (21 years or older) majoring in computer science that has recently completed 24 credits at York University and maintained a 5.00 (C+) or higher average.
- Sally Murray Findley Memorial Scholarship – a cross-Faculty scholarship available to ITEC, MATH and EECS students who have completed at least 48 credits including at least 18 credits in the major with a minimum GPA of 7.00 (B+).

Advising

General Academic advising is available on an individual basis in the Lassonde Student Welcome and Support Centre in 105 Bergeron Centre for Engineering Excellence (BCEE). Individual advising is available to students in order to discuss academic issues such as recommended mathematical skills, theoretical versus applications oriented courses, areas of specialization, graduate studies and career paths, course choice, assistance with degree program checklists and requirements. Specialised Advising regarding the six degree programs offered by EECS is available in the Lassonde Building 1012M. This in-department advising may involve discussion and follow-up on unfavourable degree audits received by students from the Registrar's Office, special permissions, as well as elaboration on actions taken by the department (approvals for major/degree/Faculty transfers, results of the waiting list exercise, results of the prerequisites audit exercise).

It is ultimately the responsibility of each student to ensure that they meet all degree requirement aspects at the Department (major or minor requirements) level as well as at the home Faculty (i.e., Lassonde School of Engineering) and Degree levels. Written information and program checklists are provided to assist students in making appropriate choices. It is recommended that *students take advantage of advising opportunities to receive answers to any questions they may have*.

Individual advising appointments to meet with the undergraduate Vice Chairs are made through the Undergraduate Office (ug@cse.yorku.ca, Tel: (416) 736-5334).

Course Weights

Most courses meet for three class hours a week for one term (these are 3-credit courses whose numbers end in "3.00"). Many courses have required supervised labs or tutorials every week (e.g., EECS1011 3.00 and EECS1022 3.00). Some courses have labs of sufficient duration per week to entail a 4.00-credit weight for the course (e.g., EECS2021, 2602, 3201, and all 36xx and 46xx courses have a 4.00-credit weight). Catalogue numbers are assigned to the labs or tutorials rather than the lectures and students use the *Registration and Enrolment Module* (REM) to enrol by selecting an appropriate lab or tutorial. All EECS courses put heavy demands on the student's time by requiring the completion of take-home assignments or projects.

Definition of Core Courses

These are courses required in *all* degree programs in Computer Science and Computer Security. The core computer science courses are EECS1001, EECS1019, EECS1012, EECS1022, EECS2030, EECS2001, EECS2011, EECS2021, EECS2031, EECS3101, and EECS3311. Core mathematics courses are MATH1300 3.00, MATH1310 3.00, and MATH1090 3.00.

Service Courses

Service courses are courses that the department mounts for the needs of other majors, e.g., Mathematics and Statistics, Psychology, Biology, Physics, Chemistry. We identify such courses by assigning them a second digit 5 (e.g., 1520, 1530, 1540, 1541, 1550, 1560, 1570, 2501). They cannot be taken as EECS major credits[1] and grades in them are not included in calculating the "EECS grade point average" that the general prerequisites speak of (see above under "Prerequisites").

Prerequisites

Almost all EECS courses have prerequisites. These are carefully considered in order to provide accurate information to students about what background they need to have before taking the course. *The department does not permit students to continue without the prerequisites in any EECS major course*.

Students are encouraged to familiarise themselves with the prerequisites of each course they are interested in enrolling, either in this publication or via the *advanced course search* on York University's "Search for Courses" web pages for current students.

[1] *There is an exception to this: The 2013 electrical engineering cohort was required to take EECS 1541 3.00 and as a result, for this cohort only, we count this course as a major course* ***and*** *include the grade obtained in the computation of the EECS gpa.*

Prerequisites are enforced in every term via a prerequisite audit process undertaken by the Undergraduate Office. *Independently of their major*, students who are identified during the audit as being enrolled in EECS courses for which they do not meet the prerequisite *will be de-enrolled and notified by email*. This prerequisite auditing process starts as early as possible after the start of each term and, depending on Undergraduate Office workload, may continue up to the end of the sixth week of the term.

Many EECS courses at levels 2000-4000 include "general prerequisites" as one of their prerequisites. General prerequisites means a cumulative grade point average of 4.5 or higher over all major EECS courses. Major EECS courses include courses like EECS 1019, EECS 1028, EECS 3121, EECS 3122 and EECS 4161, even if they appear on your transcript as MATH courses. EECS courses for non-majors (i.e., those with second digit 5) are not included in the gpa calculation. Co-op and internship courses are also excluded from the gpa calculation.

In exceptional circumstances, some prerequisites or corequisites may be waived at the discretion of the undergraduate director in consultation with the course director, when the student has equivalent academic preparation. Course directors may not waive prerequisites.

Note. A semicolon in a prerequisite list is synonymous with **"and".**

Course Credit Exclusions

Two courses are "Course Credit Exclusions", or CCE, if they have such degree of overlap that they cannot both count for credit. By Senate policy it is the chronologically second of the two that counts, while the first is flagged "NCR" (no credit retained). CCE does not imply that one course is acceptable in lieu of the other in degree requirements.

Access to Courses

York Enrolment System

Students enrol in courses using the Registration and Enrolment Module (**REM**), via a Web interface, typically in the few months prior to the start of each term. EECS courses occasionally reach their class size maximum, in which case the following procedures are followed. (See http://eecs.lassonde.yorku.ca/current-students/undergrads-courses/requirements-faq/enrollment-guides/ for an expanded description and interpretation of the enrolment policy outlined below.)

Engineering Sections

To meet accreditation standards, some of the courses taken by B.Eng. students must be taught by professional engineers. In multi-section courses, the sections for engineers are identified by section names E and Z. B.Eng students *must* enrol in an E or Z section. Consequently, seats in these sections may be reserved for engineering students. Other students may be admitted to them if space permits.

Waiting List

We are committed to ensuring that students majoring in Computer Science, Computer Security, Digital Media, Computer Engineering, Software Engineering and Electrical Engineering can make *timely progress* towards meeting their degree requirements.

Near the beginning of each term, students who were not able to enrol in EECS will have the opportunity to apply for the waiting list, using an online form that is available via the departmental web page. The department processes these applications, giving priority to students who need the course for normal progress. When results of the waiting list are posted, students who have been granted permission to enrol in particular courses may then enrol themselves in the courses.

Students who wish to take more EECS courses than they need to—for example, to accelerate their studies—or who wish to repeat a course that they either dropped or in which they obtained unsatisfactory grades in a preceding term, *will only be accommodated if space permits*.

Normal progress is consistent with completion times of four and three years for full-time students in the Honours and Bachelor degree programs respectively. This entails:

- Taking up to a total of three EECS courses per term (four in the cases of engineering majors due to their heavier degree requirements) that the prerequisite structure permits.
- When close to graduation, being able to take necessary courses within the limits specified above.

Limits on Course Enrolment

A maximum combined number of **three** (**four** for engineering) EECS courses are permitted in any given fall or winter term, subject to prerequisites being met. *In the summer term students are not permitted to take more than a maximum of* ***two*** *EECS courses*.

If any student enrols in more than the allowed number of courses per term they will be de-enrolled *from courses in which the Department requires space*.

Academic Honesty

As students in a professional school, you are expected to maintain the highest standards of academic integrity and honesty. The University Senate, the Lassonde School of Engineering and the Department have policies on academic honesty and their enforcement is taken very seriously. Academic honesty is essentially giving credit where credit is due. When a student submits a piece of work it is expected that all unquoted and unacknowledged ideas (except for common knowledge) and text are original to the student. Unacknowledged and unquoted text, diagrams, etc., which are not original to the student, and which the student presents as their own work is *academic dishonesty*. The deliberate presentation of part of another student's program text or other work as your own without acknowledgment is academically dishonest, and renders the student liable to the disciplinary procedures instituted by Senate.

The above statement does not imply that students must work, study and learn in isolation. The Department encourages students to work, study and learn together, and to use the work of others as found in books, journal articles, electronic news and private conversations. In fact, most pieces of work are enhanced when relevant outside material is introduced. Thus, faculty members expect to see quotes, references and citations to the work of others. This shows the student is seeking out knowledge and integrating it with their work.

As long as appropriate citation and notice is given, students cannot be accused of academic dishonesty.

A piece of work, however, may receive a low grade because it does not contain a sufficient amount of original work. In each course, instructors describe their expectations regarding cooperative work and define the boundary of what is acceptable cooperation and what is unacceptable. When in doubt, it is the student's responsibility to seek clarification from the instructor. Instructors evaluate each piece of work in the context of their course and given instructions.

You should refer to the appropriate sections of the York University Undergraduate Calendar http://calendars.registrar.yorku.ca and Senate policy

http://www.yorku.ca/secretariat/policies/document.php?document=69 for further information about academic honesty and the penalties when academic dishonesty occurs.

Grading System

Grading at York University is done on a letter scale. See http://www.yorku.ca/secretariat/policies/document.php?document=87

The following table shows the grading scale used. The number in parentheses is the grade point that is used to determine the grade point average. The grade point average is a credit weighted average of all relevant courses.

- A+ (**9) Exceptional**. Thorough knowledge of concepts and/or techniques and exceptional skill or great originality in the use of those concepts, techniques in satisfying the requirements of an assignment or course.
- A (**8) Excellent.** Thorough knowledge of concepts and/or techniques with a high degree of skill and/or some elements of originality in satisfying the requirements of an assignment or course.
- B+ (**7) Very Good.** Thorough knowledge of concepts and/or techniques with a fairly high degree of skill in the use of those concepts, techniques in satisfying the requirements of an assignment or course.
- B (**6) Good.** Good level of knowledge of concepts and/or techniques together with considerable skill in using them to satisfy the requirements of an assignment or course.

- C+ (**5) Competent.** Acceptable level of knowledge of concepts and/or techniques together with considerable skill in using them to satisfy the requirements of an assignment or course.
- C (**4) Fairly Competent**. Acceptable level of knowledge of concepts and/or techniques together with some skill in using them to satisfy the requirements of an assignment or course.
- D+ (**3) Passing.** Slightly better than minimal knowledge of required concepts and/or techniques together with some ability to use them in satisfying the requirements of an assignment or course.
- D (**2) Barely Passing**. Minimum knowledge of concepts and/or techniques needed to satisfy the requirements of an assignment or course.
- E (**1) Marginally Failing**.
- F (**0) Failing**.

Concerns about Fairness

The Department's faculty members are committed to treating all students fairly, professionally, and without discrimination on non-academic grounds including a student's race or sex. Students who have concerns about fair treatment are encouraged to discuss the matter with their instructor or the course coordinator, if applicable (e.g., EECS1520 3.00). If this is not possible or does not resolve the problem, the matter should be brought to the attention of the Undergraduate Director, and if necessary, the Department Chair, for a departmental response.

Appeal Procedures

The Department expects a student's disagreement with an evaluation of an item of course work (e.g., final examination, assignment report, class test, oral presentation, laboratory presentation, class participation) to be settled with the instructor informally, amicably and expeditiously.

If however a formal appeal becomes necessary due to lack of an informal settlement, then there are distinct procedures to follow for term work on one hand and for final examinations and final grades on the other. Of necessity, a formal appeal must involve only written work.

Term Work

An appeal against a grade assigned to an item of term work must be made to the instructor *within 14 days of the grade being made available to the class.*

In the case of a multi-sectioned course (where the instructor is not the course director), a second appeal may be made to the course director *within 14 days of the decision of the instructor.*

If a student feels that their work has not been fairly reappraised by the course director, then they may appeal for a reappraisal by the Departmental petitions committee. Such a request is made in writing using the appropriate form obtained from the Undergraduate Office. The request must be made *within 14 days of the decision of the course director.*

Final Exams and Final Grades

An appeal for reappraisal of a final grade must be made in writing on a the appropriate Departmental form, obtained from the Undergraduate Office, *within 21 days of receiving notification of the grade or by the date set by the Registrar's Office.*
For more details on the University's reappraisal policies, please see http://myacademicrecord.students.yorku.ca/grade-reappraisal-policy

The Departmental petitions committee will discuss the appeal with the course director to ensure that no grade computation, clerical or similar errors have been made. If such an error is discovered, a correction will be made and the student and the Registrar's Office will be notified.

If a final examination is to be reappraised then the Departmental petitions committee will select a second reader for the examination paper. The petitions committee will consider the report of the second reader and recommend a final grade, which may be lower, the same, or higher than the original grade. The student will receive the report of the petitions committee in writing and the Registrar's Office will be informed of any grade change. The decision of the Department Petitions Committee can *only be appealed on procedural grounds* to the Petitions Committee of the Faculty.

Associated Course Fees

Courses above the 1000 level have an associated fee of $5.00, with the exception of internship and co-op courses, whose associated fees cover the cost of running the co-op programs.

The cost of these fees is reviewed from year to year and adjusted accordingly.

Courses Taken Outside York

Students wishing to take Computer Science courses at another university should submit a **Letter of Permission** (**LOP**) to the University (on line: http://www.registrar.yorku.ca/enrol/lop). For the purpose of satisfying degree requirements, the number of computer science course (EECS courses) credits taken outside the Department of Electrical Engineering and Computer Science may not exceed 6 credits in ***core*** computer science courses, and 12 EECS credits in total, *for the duration of the student's program of study*. *Transfer Credit assessed at the point of Admission is included as credit taken outside the Department*.

Moving to New Program Requirements and Prerequisites

Our disciplines constantly respond and adapt to technological and theoretical progress. To ensure that our programs are informed by the latest advances in the field, the Department has determined the following principles governing the applicability of new degree requirements for Computer Science programs:

- If you have been taking courses in consecutive years, then the starting year in a computer science (computer security, digital media) major is the year in which you take your first major EECS course *as a Computer Science (or Computer Security or Digital Media) major*. This year *normally* coincides with the year you were admitted into the program, *unless you delayed taking major courses*. If you have a break of three or more consecutive terms in your studies—or you have changed your program—then your *starting year is redefined* to be the year in which you start taking major EECS courses once you come back, or, respectively, the year in which you changed your program. Since most Senate approved degree program regulations become effective in the fall term following their approval, your starting year is the current academic year if you start in the fall, winter, or the *immediately following summer term*. For example: starting in fall 2001 you follow the 2001-02 program requirements; starting in winter 2002 or summer 2002 you also follow the 2001-02 program requirements.

- If program requirements change but you had neither an interruption in your studies, nor a change in programs, then you may continue with your studies using the program requirements in effect in your starting year. In this case the degree checklists in this calendar may not apply to you. You should use the degree checklists applicable to your starting year. You may find these at this link:

  http://eecs.lassonde.yorku.ca/current-**students**/undergrads-courses/eecs-supplemental-calendars/
- If program requirements change you may elect to graduate under the *new* requirements—that is, those in effect in the year of your graduation—but you must meet all of them. You are not permitted to mix and match old and new requirements, or to pick and choose from among various requirements that were in effect between your starting year and graduation year.
- Changes in prerequisites for courses are not changes in degree requirements, and apply to all students regardless of their year of entry or re-entry to the program. In fact, being “course-based” rather than “degree-based” requirements, prerequisites apply to majors and non-majors alike. Prerequisite changes are normally effective starting with the term immediately following their approval by Faculty Council.

Course Descriptions: 1000-Level

EECS 1001 1.00 Research Directions in Computing

Computer Science is an exciting and wide-ranging discipline, many of whose topics will not be introduced in any technical depth until upper year courses. This course consists of a set of invited lectures by researchers in the department and a set of other organised events that will introduce the students to the breadth of computer science.

The course is organised around a series of invited talks by individual researchers and research groups, as well as a number of laboratory tours and other events that will introduce students to specific research directions in computer science, issues related to professionalism and professional societies, and opportunities to become engaged in different research and technical groups and events related to computer science.

Formally, the course consists of 12 one-hour lectures spread over two terms. The first lecture will be organizational in nature. The remaining 11 lectures are comprised of invited lectures by researchers (or research groups) in computer science, representatives of specific interest groups associated with computer science (e.g., Engineers Without Borders, Canadian Information Processing Society, etc.), work-study/internship/student exchange programs, and representatives of volunteer/other organizations that seek out technically literate students as volunteers.

In addition to these 12 formal lectures, a set of other extracurricular events will be organised including research lab tours, visits to local industrial sites (e.g., IBM), special lectures directed at specific technical problems often encountered by students (e.g., running LINUX at home), etc.

This course is offered on a pass-fail basis only.

Note. Computer Science and Computer Security Majors are expected to complete this course in their first year of study.

EECS 1011 3.00 Computational Thinking through Mechatronics

The objectives of this course are threefold: providing a first exposure to procedural programming, teaching students a set of soft computing skills (such as reasoning about algorithms, tracing programs, test-driven development), and demonstrating how computers are used in a variety of engineering disciplines. It uses problem-based pedagogy to expose the underlying concepts and an experiential laboratory to implement them. An integrated computing environment (such as MATLAB) is used so that students can pick up key programming concepts (such as variables and control flow) without being exposed to complex or abstract constructs. The problems are chosen in consultation with the various engineering disciplines in the Faculty with a view of exposing how computing is used in these disciplines. The lectures (two hours weekly) are supplemented by a three-hour weekly lab.

Learning outcomes for the course:

- Use a set of soft computing skills such as reasoning about algorithms, tracing programs, and test-driven development for programming applications.
- Explain and apply the fundamental constructs in procedural programming, including variables and expressions, control structures (conditionals/loops), and documentation.
- Write simple programs using functions defined in m-files.
- Use the computing environment to implement/simulate selected applications from science, math, and engineering.

Prerequisites: None

Course Credit Exclusions: EECS1541 3.00

EECS 1012 3.00 Introduction to Computing: A Net-centric Approach

The objectives of EECS 1012 are threefold: providing a first exposure to event-driven programming, teaching students a set of computing skills (including reasoning about algorithms, tracing programs, test-driven development, unit testing), and providing an introduction to computing within a mobile, net-centric context. It uses a problem-based approach to expose the underlying concepts and an experiential laboratory to implement them. A mature mobile software infrastructure (such as HTML, CSS, and JavaScript) is used so that students can pick up key programming concepts (such as variables and control flow) within a client-server context without being bogged down in complex or abstract constructs. Laboratory exercises expose students to a range of real-world problems with a view of motivating computational thinking and grounding the material covered in lecture. The lectures (two hours weekly) are supplemented by a three-hour weekly lab.

Learning outcomes for the course:

- Use a set of computing skills such as reasoning about algorithms, tracing programs, test-driven development, and diagnosing faults.
- Explain and apply fundamental constructs in event-driven programs, including variables and expressions, control structures (conditionals/loops), and API usage.
- Write simple programs using a given software infrastructure, API, and tool chain.

- Gain exposure to net-centric computing, client-server applications, and simple relational database use.
- Become familiar with the notion of syntax, both for programs and documents, and the principle of separation of concerns.

Prerequisites: One of (1)—(3) below must be met:

(1) (New high school curriculum): One 4U Math course with a grade of at least 75%;
(2) Completion of 6.00 credits from York University MATH courses (not including courses with second digit 5) with a grade point average of 5.00 (C+) or better over these credits;
(3) Completion of 6.00 credits from York University mathematics courses whose second digit is 5, with an average grade not below 7.00 (B+).

Course Credit Exclusions: AP/ITEC3020 3.00, SC/CSE2041 3.00, LE/SC/CSE2041 4.00, LE/EECS2041 4.00.

EECS 1019 3.00 Discrete Mathematics for Computer Science (Cross-listed with MATH 1019 3.00)

Introduction to abstraction; use and development of precise formulations of mathematical ideas; informal introduction to logic; introduction to naïve set theory; induction; relations and functions; big-O notation; recursive definitions, recurrence relations and their solutions; graphs and trees.

Prerequisites: SC/MATH1190 3.00, or GL/MATH 1650 3.00, or **two** 4U courses including MHF4U (Advanced Functions)
Course Credit Exclusions: SC/MATH2320 3.00, LE/EECS1028 3.00, SC/MATH1028 3.00

EECS 1021 3.00 Object Oriented Programming from Sensors to Actuators

The objective of this course is to introduce computational thinking—a process-based approach—to problem solving. It uses a problem-based pedagogy to expose the underlying concepts and an experiential laboratory to implement them. The programming language is chosen so that it is widely used in a variety of applications, is object-oriented, and is of industrial strength (Java is an example of such a language). The problems are chosen in order to expose abstract programming concepts by immersing them in relevant and engaging applications. The experiential laboratory is based on sensors and actuators that connect to a computer. The problems are chosen in consultation with the various engineering disciplines in the Faculty with a view of exposing how computing is used in these disciplines.

The lab hardware is chosen so that it can interface with a variety of languages (including MATLAB and Java).

Learning outcomes for the course:

- Demonstrate the ability to test and debug a given program and reason about its correctness.
- Given a problem specification and a suitable API, build an application that meets the given requirement.
- Use ready-made collections to solve problems involving aggregations of typed data.
- Build an event-driven application that controls sensors and actuators in order to connect events to physical actions.
- Program common applications from a variety of engineering disciplines using an object oriented language and solve them on the computer.

Prerequisites: LE/EECS1011 3.00

Course Credit Exclusions: LE/EECS1022 3.00, LE/EECS1020 3.00, LE/CSE1020 3.00, AK/AS/SC/CSE1020 3.00, AP/ITEC1620 3.00

EECS 1022 3.00 Programming for Mobile Computing

This course provides a first exposure to object-oriented programming and enhances student understanding of key computing skills such as reasoning about algorithms, designing user interfaces, and working with software tools. It uses a problem-based approach to expose the underlying concepts and an experiential laboratory to implement them. A mature mobile software infrastructure (such as Java and the Android programming environment) is used to expose and provide context to the underlying ideas. Laboratory exercises expose students to a range of real-world problems with a view of motivating computational thinking and grounding the material covered in lectures.

Learning outcomes for the course:

- Understand software development within an object-oriented framework using a modern programming language and tool set.
- Use a set of computing skills such as reasoning about algorithms, tracing programs, test-driven development, and diagnosing faults.
- Explain and apply fundamental constructs in event-driven programs, including variables and expressions, control structures (conditionals/loops), and API usage.
- Write simple programs using a given software infrastructure, API, and tool chain.
- Gain exposure to a comprehensive mobile computing framework.
- Gain exposure to user interface design.

Prerequisites: LE/EECS1012 3.00

Course Credit Exclusions: LE/EECS1021 3.00, LE/EECS1020 3.00, LE/CSE1020 3.00, AK/AS/SC/CSE1020 3.00, AP/ITEC1620 3.00

EECS 1028 3.00 Discrete Mathematics for Engineers
(Cross-listed with MATH1028 3.00)

Introduction to abstraction; use and development of precise formulations of mathematical ideas, in particular as they apply to engineering; introduction to propositional logic and application to switching circuits; sets, relations and functions; predicate logic and proof techniques; induction with applications to program correctness; basic counting techniques; graphs and trees with applications; automata and applications. Weekly three-hour lectures and two hours of mandatory tutorials per week.

Learning outcomes for the course:

- Prove or disprove any propositional formula as the case requires, using truth tables or syntactic proof techniques such as resolution.
- Prove or disprove as the case may be simple formulas in quantified logic.
- Translate English mathematical statements into predicate logic formulas.
- Prove simple mathematical statements by contradiction, by cases, or by assuming the antecedent.

- Prove by induction statements that depend on a natural number; in particular: Prove the correctness of single loop programs and simple recursive programs.
- Prove statements about inductively defined objects by structural induction; in particular: Prove the correctness of simple recursive programs.
- Be able to reason about graphs and (binary) trees and use them in several examples, e.g., demonstrate an ability to locate the fundamental cycles of an electrical circuit
- Be able to show simple properties of trees (examples: relation between number of nodes and edges; relation between number of nodes and height)
- Construct automata that can recognize in a text its "arbitrary" words and its specific "keywords" the latter according to a given list (corresponding to the action of a "scanner" module in a compiler)
- Construct simple automata to assess the correctness of simple concurrent programs.

Prerequisites: MHF4U and MCV4U

Course Credit Exclusions: LE/EECS1019 3.0, LE/SC/CSE1019 3.0, SC/MATH1019 3.00, SC/MATH2320 3.00

EECS 1520 3.00 Computer Use: Fundamentals

This course is appropriate for students who are not majoring in engineering or computer science, but who would like an introduction to the use of the computer as a problem-solving tool. No previous computing experience is assumed, but the course does involve extensive practical work with computers, so some facility with problem-solving and symbolic operations will be very helpful.

An introduction to the use of computers focusing on concepts of computer technology and organization (hardware and software), and the use of applications and information retrieval tools for problem solving.

Topics to be studied include: the development of information technology and its current trends; analysis of problems for solution by computers, report generation, file processing; spreadsheets; databases; numeric and symbolic calculation; the functions of an operating system; interactive programs.

Students should be aware that, like many other computer courses, this course is demanding in terms of time, and should not be added to an already heavy load. There is scheduled and unscheduled time in the Glade laboratory. The course is not appropriate for students who want more than an elementary knowledge of computing and it **cannot** be used as a substitute for major EECS courses such as EECS1012, EECS1020, and EECS1022.

Prerequisites: None

NCR Note: No credit will be retained if this course is taken after the successful completion of or simultaneously with LE/EECS1020 3.00, LE/CSE1020 3.00, AK/AS/SC/CSE1020 3.00, LE/EECS1021 3.00, LE/EECS1022 3.00

Note: This course counts as elective credits towards satisfying Faculty degree requirements but does not count as major credits in any of the EECS programs.

EECS 1530 3.00 Computer Use: Programming

Concepts of computer systems and technology — e.g., software engineering, algorithms, programming languages and theory of computation are discussed. Practical work focuses on problem solving using a high-level programming language. The course requires extensive laboratory work.

Note: This course is designed for students who are *not* majoring in any one of the EECS programs. However, those who wish to major in such a program but lack programming background may use it as preparation. This course does not count as a major credit in any program offered by EECS.

Prerequisites: None

Course Credit Exclusions: LE/CSE1530 3.00, AK/AS/SC/CSE1530 3.00, LE/EECS1540 3.00, LE/CSE1540 3.00, AK/AS/SC/CSE1540 3.00

NCR Note: No credit will be retained if this course is taken after the successful completion of or simultaneously with LE/EECS1020 3.00 or LE/SC/CSE1020 3.00 or LE/EECS1021 3.00 or LE/EECS1022 3.00 or AP/ITEC1620 3.00

EECS 1540 3.00 Computer Use for the Natural Sciences

Introduction to problem solving using computers — top down and modular design; implementation in a procedural programming language — control structures, data structures, subprograms; application to simple numerical methods, modelling and simulation in the sciences; use of library subprograms. This course is intended for students in the Faculty of Science.

Prerequisites: None.

Course Credit Exclusions: LE/CSE1540 3.00, AK/AS/SC/CSE1540 3.00, LE/EECS1530 3.00, LE/CSE1530 3.00, AK/AS/SC/CSE1530 3.00

NCR Note: No credit will be retained if this course is taken after the successful completion of or simultaneously with LE/EECS1020 3.00, LE/SC/CSE1020 3.00 or LE/EECS1021 3.00 or LE/EECS1022 3.00 or AP/ITEC 1620 3.00

EECS 1541 3.00 Introduction to Computing for the Physical Sciences

This course introduces students to computer-based problem solving techniques that can be used to approach problems in the physical sciences, such as answering questions that require numerical computation, as well as basic analysis of experimental data sets and simple statistical simulations.

Prerequisites: SC/MATH1013 3.00 or equivalent.

Co-requisites: SC/PHYS1010 6.00 or SC/PHYS1410 6.00 or SC/PHYS1420 6.00; and SC/MATH 1021 3.00 or SC/MATH 1025 3.00

Course Credit Exclusions: LE/EECS1560 3.00, LE/SC/CSE1560 3.00, LE/EECS1570 3.00, LE/SC/CSE1570 3.00

EECS 1550 3.00 Introduction to Web Development (not offered in 2018-2019)

This is an introduction to the development of interactive web applications. Topics include Hyper Text Markup Language (HTML), which is the language used to create web pages, Cascading Style Sheets (CSS), which is used to specify the visual style used to display web pages, and

JavaScript, which is a programming language used to create dynamic and interactive web pages.

Learning outcomes for the course:

- Create web pages using Hyper Text Markup Language (HTML) and Cascading Style Sheets (CSS).
- Use JavaScript to create interactive web pages.
- Use the Internet to find and make use of information related to the course topics.

Prerequisites: None

NCR Note: No credit will be retained if this course is taken after the successful completion of, or simultaneously with LE/EECS 1012 3.00, GL/ITEC 2635 3.00 or AP/ITEC3020 3.00

EECS 1560 3.00 Introduction to Computing for Math and Statistics

This course introduces students to computer-based problem solving techniques that can be used to approach problems in Mathematics and Statistics using the MATLAB programming language. Through a combination of lectures and laboratory sessions, students become familiar with a scientific computing environment that combines numeric computation, data handling, high-level programming, graphics, and a variety of visualization tools.

Prerequisites: SC/MATH1300 3.00
Co requisites: SC/MATH1310 3.00; SC/MATH1131 3.00 or SC/MATH 2030 3.00
Course Credit Exclusions: LE/EECS1541 3.00, LE/SC/CSE1541 3.00, LE/EECS1570 3.00, LE/SC/CSE1570 3.00
NCR Note: No credit will be retained if this course is taken after the successful completion of, or simultaneously with SC/PHYS2030 3.00

EECS 1570 3.00 Introduction to Computing for Psychology

This course introduces students to computer-based problem solving techniques that can be used to approach problems in psychology such as the design of stimulus-response experiments and the capture and simple analysis of data from a variety of experimental contexts. The analysis of data will mainly focus on data management such as how to deal with files that come in different formats, how to make new variables, how to make subsets of files, how to combine files, how to ftp files, etc. Through a combination of lectures and weekly exercises students learn the basic concepts of computer programming with application to such a domain. In addition to an in-depth focus on one programming environment the course provides an overview of a range of other experimental environments used in psychology including brief exposure to a statistical analysis package. This brief exposure will not go beyond very basic descriptive statistics and creation of graphs.

Faculty from the Department of Psychology will participate in developing domain-specific lab exercises. The course is a lecture-based course (3 hours per week) with an extensive component of weekly exercises through which student “learn by doing”.

Prerequisite: SC/MATH1505 6.00
Course Credit Exclusions: LE/EECS1541 3.00, LE/EECS1560 3.00, LE/SC/CSE1541 3.00, LE/SC/CSE1560 3.00

EECS 1710 3.00 Programming for Digital Media

The course lays the conceptual foundation for the development and implementation of Digital Media artefacts and introduces some of the core concepts of Digital Media, including the computing and cultural layers of media, and the notion of cultural logic (Media Theory). Topics include programming constructs, data types and control structures; the object oriented concepts of modularity and encapsulation; integration of sound, video, and other media; networking constructs (HTTP connections); and the interrelationships among languages such as Processing, Java, and other Digital Media tools (such as Macromedia Director and Python).

This course is an introduction to the interdisciplinary area of practice of New Media; it is not a survey course. As such, the emphasis is on the development of a theoretical conceptual foundation and the acquisition of the intellectual and practical skills required for further courses in the Digital Media program, and thus is intended for prospective majors in this program. It is not intended for those who seek a quick exposure to Digital Media, or Digital Media applications or programming.

Prerequisites: None.
Course Credit Exclusions: LE/EECS1530 3.00, LE/SC/CSE1530 3.00, AP/ITEC1620 3.00
NCR Note: No credit will be retained if this course is taken after the successful completion of, or simultaneously with LE/EECS1021 3.00 or LE/EECS1022 3.00 or LE/EECS1020 3.00 or LE/SC/CSE1020 3.00

EECS 1720 3.00 Building Interactive Systems

This course continues an introduction to computer programming within the context of image, sound and interaction, subsequent to EECS1720 3.00. The student's foundation in basic programming will serve as a platform from which to explore the use of diverse media within interactive systems, including the WWW and simple game systems.

Prerequisites: LE/EECS1710 3.00
Course Credit Exclusions: LE/EECS1020 3.00, LE/SC/CSE1020 3.00, LE/EECS1021 3.00, LE/EECS1022 3.00, AP/ITEC1620 3.00, AP/ITEC1630 3.00

EECS 1910 3.00 Industry Practicum and EECS 1911 3.00 Industry Practicum

These experiential education courses are available only to students in the Industry Partnership Stream of the Computer Science Honours BSc. Students in the stream may enroll during a term in which they are employed by the industrial partner. Personalized learning outcomes for the course will be developed by the course instructor in consultation with the student's industry mentor prior to enrolment.

Prerequisites: none.

Course Descriptions: 2000-Level

EECS 2001 3.00 Introduction to the Theory of Computation

The course introduces different theoretical models of computers and studies their capabilities and theoretical limitations. Topics covered include finite automata and regular expressions; practical applications, e.g., text editors; Pushdown automata and context-free grammars;

practical applications, e.g., parsing and compilers; and Turing machines as a general model of computers, introduction to unsolvability, the halting problem.

Learning outcomes for the course:

- Design machines (i.e., finite automata, Turing machines) to solve specified decision problems.
- Design regular expressions and context-free grammars for a given language.
- Explain why an object designed in bullets (1) or (2) correctly meets its specification.
- Prove simple properties about models of computation (e.g., that the class of regular languages is closed under complement).
- Demonstrate limits of computing by proving that a problem is not solvable within a particular model of computation.
- Show how one problem can be reduced to another.

Prerequisites: General prerequisites; LE/EECS1021 3.00 or LE/EECS1022 3.00 or LE/EECS1720 3.00 or LE/EECS1030 3.00; LE/EECS1028 3.00 or SC/MATH1028 3.00 or LE/EECS1019 3.00 or SC/MATH1019 3.00

EECS 2011 3.00 Fundamentals of Data Structures

This course discusses the fundamental data structures commonly used in the design of algorithms. At the end of this course, students will know the classical data structures, and master the use of abstraction, specification and program construction using modules. Furthermore, students will be able to apply these skills effectively in the design and implementation of algorithms.

Learning outcomes for the course:

- Instantiate a range of standard abstract data types (ADT) as data structures.
- Implement these data structures and associated operations and check that they satisfy the properties of the ADT.
- Apply best practice software engineering principles in the design of new data structures.
- Demonstrate the ability to reason about data structures using contracts, assertions, and invariants.
- Analyse the asymptotic run times of standard operations for a broad range of common data structures.

- Select the most appropriate data structures for novel applications.

Prerequisites: General prerequisites; LE/EECS1030 3.00 or LE/EECS2030 3.00; LE/EECS1028 3.00 or SC/MATH1028 3.00 or LE/EECS1019 3.00 or SC/MATH1019 3.00

EECS 2021 4.00 Computer Organization

This course provides a description of how computers work by following the abstraction trail from the high-level programming layer down to the digital-logic component layer. By understanding how the features of each abstraction layer are implemented in the one beneath it, one can grasp the tapestry of the software/hardware interface.

Topics include programming in assembly language, machine instructions and their encoding formats, translating and loading high-level programs, computer organization and performance issues, CPU structure, single/multi-cycle datapath and control, pipelining, and memory hierarchy. The course presents theoretical concepts as well as concrete implementations on a modern RISC processor.

Learning outcomes for the course:

- Translate high-level code to assembly language and machine code.
- Represent data in machine readable form and describe how it is stored and manipulated in a CPU.
- Synthesize hardware of increasing complexity from logic gates to a simple CPU using a Hardware Description Language.
- Evaluate computer performance and compare performance on different architectures and designs.
- Describe and critique I/O and Parallel Hardware.

Prerequisites: General prerequisites; LE/EECS1021 3.00 or LE/EECS1022 3.00 or LE/EECS1720 3.00 or LE/EECS1030 3.00

EECS 2030 3.00 Advanced Object Oriented Programming

This course continues the separation of concern theme introduced in its predecessors EECS1720, EECS1021 and EECS1022. While those courses focus on the client concern, this course focuses on the concern of the implementer. Hence, rather than using an API (Application Programming Interface) to build an application, the student is asked to implement a given API. Topics include implementing classes (utilities/nonutilities, delegation within the class definition, documentation and API generation, and implementing contracts), aggregations (implementing aggregates versus compositions and implementing collections), inheritance hierarchies (attribute visibility, overriding methods, abstract classes versus interfaces, inner classes); generics; building graphical user interfaces with an emphasis on the MVC (Model-View-Controller) design pattern; recursion; searching and sorting (including quick and merge sorts); linked lists; and stacks and queues. The coverage also includes a few design patterns. Throughout the course an IDE (Integrated Development Environment), such as eclipse, and a testing framework, such as JUnit, are used.

Learning outcomes for the course:

- Implement an API (Application Programming Interface).
- Test the implementation.
- Document the implementation.
- Implement aggregations and compositions.
- Implement inheritance.
- Use recursion.
- Implement linked lists.
- (Informally) prove that recursive algorithms are correct and terminate.
- (Informally) analyse the running time of (recursive) algorithms.

Prerequisites: General prerequisites; LE/EECS1021 3.00 or LE/EECS1020 3.00 or LE/EECS1022 3.00 or LE/EECS1720 3.00
Course Credit Exclusions: LE/EECS1030 3.00, AP/ITEC2620 3.00

EECS 2031 3.00 Software Tools

This course introduces software tools that are used for building applications and in the software development process. It covers ANSI-C (stdio, pointers, memory management, overview of ANSI-C libraries), Shell programming including Filters and pipes (shell redirection, grep, sort & uniq, tr, sed, awk, pipes in C), Version control systems and the "make" mechanism, and debugging and testing. All of the above are applied in practical programming assignments and/or small-group projects.

Learning outcomes for the course:

- Use the basic functionality of the Unix shell, such as standard commands and utilities, input/output redirection, and pipes.

- Develop and test shell scripts of significant size.
- Develop and test programs written in the C programming language.
- Describe the memory management model of the C programming language.
- Use test, debug and profiling tools to check the correctness of programs.

Prerequisites: General prerequisites; one of LE/EECS 1021 3.00 or LE/EECS 1022 3.00 or LE/EECS1030 3.00

EECS 2200 3.00 Electrical Circuits

This course covers the basic principles of linear circuits. Kirchhoff's laws, circuit equations, RL, RC, and RLC circuits, three-phase circuits, power analysis and power factor, and magnetically coupled circuits. Topics covered include circuit elements, current and voltage sources, power, Kirchhoff's laws, dependent sources, Nodal analysis, mesh analysis, operational amplifiers, Superposition, Thevenin's and Norton's theorems, inductance capacitance and duality, First order RL and RC circuits, Second order linear circuits, sinusoidal steady state analysis, sinusoidal steady state power calculation, power factor and correction, introduction to Laplace transform, and magnetically coupled circuits and transformers.

Learning outcomes for the course:

- Analyse resistive circuits using basic laws (node and loop analysis, Thevenin, Norton's, superposition).
- Determine the transient response of RC, RL, and RLC circuits.
- Analyse AC circuits in steady state using the phasor method.
- Use basic tools such as MATLAB and SPICE for circuit analysis.
- Measure basic electrical signals using electronic measurement equipment in a lab setting.

Prerequisites: General prerequisites; SC/PHYS1010 6.00 or SC/PHYS1801 3.00
Course credit exclusion: SC/PHYS3050 3.00

EECS 2210 3.00 Electronic Circuits and Devices

This course covers the basic material required in the design of both analog and digital electronic circuits. Topics covered include: Amplifiers: signals and frequency spectrum of signals, models for amplifiers, and frequency response of amplifiers. Operational Amplifiers: Inverting and non-inverting configuration, difference amplifiers, integrators and differentiators, Imperfections, frequency response and effect of finite open loop gain. Diodes: review on the basic characteristics of the diode, small signal model, Zener diode, half wave, full wave, and bridge rectifiers. SPICE diode model. Bipolar Junction Transistors: Basic characteristics and operation, small signal model, high frequency model, BJT as an amplifier (common emitter and common collector), and SPICE BJT model. MOSFET: Device structure and operation, DC circuits, small signal model, high frequency model, MOS amplifiers, CMOS, SPICE MOSFET model.

Learning outcomes for the course:

- Design and analyse basic electronic circuits that contain diodes, operational amplifiers, MOSFET and bipolar transistor.
- Use CAD tools for the analysis of electronic circuits.
- Write concise, coherent, and grammatically correct materials that reflect critical analysis and synthesis.
- Effectively work in a team to achieve group goals and contribute to effective working relationships.

Prerequisites: General prerequisites; LE/EECS2200 3.00
Course credit exclusion: SC/PHYS3150 3.00

EECS 2311 3.00 Software Development Project
This course allows students to develop their first significant piece of software. There will be formal instruction during the lecture hours providing guidance throughout the software development process on topics such as eliciting user requirements, system specification, use of modern IDEs, application development (including graphical user interfaces), unit, component, integration, and acceptance testing, as well as deployment strategies and user documentation.

However, the main intent of the course is for students to experience during the supervised lab portion the issues that engineering large software systems entails, such as changing or ambiguous requirements, understanding code written by someone else, flexible vs. inflexible design, testing adequacy, and maintainability concerns. In this way, the requirements elicitation techniques, development methodologies, design guidelines, and testing approaches presented in third and fourth year courses will be more meaningful since they will be grounded in personal experience.

The end deliverable will be judged both with respect to the quality of the user experience it provides (correctness / robustness / user-friendliness), as well as in terms of the quality of the produced software (readability / design / maintainability).

Learning outcomes for the course:

- Describe the requirements of a large software system.
- Select appropriate system elements for a high-level design description.
- Derive and implement test cases at the unit and system level.
- Produce a detailed user manual for an interactive system.
- Implement a large software system from scratch.

Prerequisites: General prerequisites; LE/EECS 1030 3.00 or LE/EECS 2030 3.00

EECS 2501 1.00 Fortran and Scientific Computing

This course covers computer-based problem solving in a variety of scientific and engineering settings. It introduces the FORTRAN programming language and its interface with scientific libraries.

The first third of the course (4 weeks) is in lecture format (3 hours per week) covering data types, control structures and program structure, functions and subroutines, arrays, I/O, and Errors in computations. For the remainder of the term students work on their own on various projects. Project applications are drawn mainly from the following scientific areas: Numerical methods, linear systems, curve fitting, non-linear equations, optimization, differential equations, Fourier transform. Simulation: random numbers, distributions, queues. Monte Carlo method. Processing experimental data. Data visualization. Chaos and fractals.

Learning outcomes for the course:

- Formulate tasks as computational problems to be solved algorithmically.
- Implement algorithms for solving engineering problems in FORTRAN.
- Test and debug programs.
- Incorporate use of library routines into FORTRAN programs.
- Apply simple numerical methods to solve problems.

Prerequisites: One of LE/EECS1020 3.00 or LE/EECS1021 3.00 or LE/EECS1022 3.00 or LE/EECS1530 3.00

EECS 2602 4.00 Signals and Systems in Continuous Time
The course introduces continuous-time (analogue) signals including an analysis and design of continuous-time systems. After reviewing core concepts in complex numbers, trigonometry, and functions, the course considers three alternate representations (differential equations, impulse response, and Laplace/Fourier transfer function) for linear, time invariant (LTI) systems in the continuous-time domain. The analysis of LTI systems is covered for each of the three representations. Frequency-selective filters are introduced as a special class of LTI systems for which design techniques based on Butterworth, Chebyshev, and Elliptic filters are covered. Applications of continuous-time systems in communications and controls are also presented. The course includes a mandatory lab that applies the theoretical concepts and algorithms learned in the course to practical, real-world applications.

Learning outcomes for the course:

- Describe a physical process in terms of signals and systems, and describe the properties of the CT systems.
- Calculate the frequency representations (Laplace and Fourier) of periodic and aperiodic CT signals.
- Compute the steady state outputs of linear time-invariant systems in the continuous-time domain using three different but equivalent techniques: (i) solving differential equations, (ii) convolution with the impulse response, and; (iii) the Fourier (or, alternatively, the Laplace) transform.
- Represent a CT linear time-invariant system using its magnitude and phase spectrum.
- Design CT frequency selective filters (in particular, the Butterworth, Chebyshev, and Elliptic filters) based on given specifications for the system.
- Analyse practical applications in controls and communication systems using the analysis techniques covered in the course.
- Represent CT signals/systems as discrete-time signals/systems and use MATLAB to analyse and design the CT signals/ systems for selected real- world applications.

Prerequisites: General prerequisites; SC/MATH1014 3.00; SC/MATH1025 3.00
Course Credit Exclusions: LE/EECS3451 4.00, LE/SC/CSE3451 4.00

Course Descriptions: 3000-Level

EECS 3000 3.00 Professional Practice in Computing

Professional, legal and ethical issues in the development, deployment and use of computer systems and their impact on society. Topics include: the impact of computing technology on society, privacy and security, computer crime, malware, intellectual property, legal issues, professional ethics and responsibilities. One third of the course will consist of guest lecturers from industry, government and the university who will typically discuss a broad range of topics related to professional issues (entrepreneurialism, small business start-up, human resources, infrastructure planning and development, research and development in industry, project management, etc.). In addition approximately another third of the course will be spent on topics related to ethics and legal issues and will usually be co-taught by faculty from a unit such as the Department of Philosophy, the Division of Social Science, or Osgoode Law School.

Learning outcomes for the course:

- Describe the main categories of ethical theories and key ways in which computer technology gives rise to new ethical issues
- Describe how computing technology affects privacy, the roles and activities of Information and Privacy Commissioners in Canada, the laws they enforce, and key international efforts (agreements and treaties) to regulate electronic data with a view to maintaining privacy
- Describe how computing technology has impacted, in both positive and negative ways, the exercise of free speech
- Describe how software is protected under copyright and patent law, how licensing is used, and how the intellectual property regime affects the development of new computing technology (both positive and negative effects)
- Describe the four criteria required for an invention to be patentable

- Search the Canadian Patent database for patents satisfying specified criteria
- Describe the key ways in which computer technology provides new challenges to law enforcement and the key elements of Canadian law that address cybercrime
- Describe the key characteristics of a profession and the role of professional organisations in establishing and upholding standards of practice and codes of conduct.

Prerequisites: General prerequisites
Course Credit Exclusions: ENG4000 6.00

EECS 3101 3.00 Design and Analysis of Algorithms
This course is intended to teach students the fundamental techniques in the design of algorithms and the analysis of their computational complexity. Each of these techniques is applied to a number of widely used and practical problems. At the end of this course, a student will be able to: choose algorithms appropriate for many common computational problems; to exploit constraints and structure to design efficient algorithms; and to select appropriate tradeoffs for speed and space. Weekly three-hour lectures and 1.5-hour scheduled mandatory tutorials.

Topics covered may include a review of fundamental data structures, asymptotic notation, solving recurrences, sorting and order statistics, divide-and-conquer approaches, dynamic programming, greedy method, divide-and-conquer algorithms, amoritization approaches, graph algorithms and the theory of NP-completeness.

Learning outcomes for the course:
- Choose an appropriate algorithm to solve a given computational problem, and justify that choice.
- Design new algorithms using a variety of techniques (recursion, greedy algorithm, dynamic programming, backtracking).
- Prove correctness of an algorithm using pre- and post-conditions and loop invariants.
- Prove bounds on the running time of an algorithm.
- Apply standard graph algorithms to a variety of problems.

Prerequisites: General prerequisites; LE/EECS2011 3.00; SC/MATH1090 3.00; SC/MATH1310 3.00

EECS 3121 3.00 Introduction to Numerical Computations I (Cross-listed with SC/MATH 3241 3.00)

This course is concerned with an introduction to matrix computations in linear algebra for solving the problems of linear equations, non-linear equations, interpolation and linear least squares. Errors due to representation, rounding and finite approximation are studied. Ill-conditioned problems versus unstable algorithms are discussed. The Gaussian elimination with pivoting for general system of linear equations, and the Cholesky factorization for symmetric systems are explained. Orthogonal transformations are studied for computations of the QR decomposition and the Singular Values Decompositions (SVD). The use of these transformations in solving linear least squares problems that arise from fitting linear mathematical models to observed data is emphasised. Finally, polynomial interpolation by Newton's divided differences and spline interpolation are discussed as special cases of linear equations. The emphasis of the course is on the development of numerical algorithms, the use of intelligent mathematical software and the interpretation of the results obtained on some assigned problems.

Prerequisites: One of LE/EECS1540 3.00 or LE/EECS2031 3.00 or SC/EECS2501 1.00; one of SC/MATH1010 3.00 or SC/MATH1014 3.00 or SC/MATH1310 3.00; one of SC/MATH1025 3.00 or SC/MATH1021 3.00 or SC/MATH2021 3.00 or SC/MATH2221 3.00

EECS 3122 3.00 Introduction to Numerical Computations II (Cross-listed with SC/MATH 3242 3.00)

The course is a continuation of EECS3121 3.00. The main topics include numerical differentiation, Richardson's extrapolation, elements of numerical integration, composite numerical integration, Romberg integration, adaptive quadrature methods, Gaussian quadrature, numerical improper integrals; fixed points for functions of several variables, Newton's method, Quasi-Newton methods, steepest descent techniques, and homotopy methods; power method, Householder method and QR algorithms.

Prerequisite: LE/EECS3121 3.00

EECS 3201 4.00 Digital Logic Design

This course covers the basic principles of switching circuit design and the design and analysis of both combinational and sequential circuits. It also introduces the students to hardware description languages and modern CAD tools. Topics covered include switching circuits, analysis and design of combinatorial circuits, hardware description languages, analysis and design of sequential circuits, timing considerations and the design of datapath and controllers.

Learning outcomes for the course:

- Analyse transistor switching circuits in terms of logic behaviour, signal levels and timing .
- Use Hardware Description Languages to design and realize standard and custom combinational and sequential circuits.
- Implement and test digital systems in programmable logic using modern CAD and test tools.
- Choose and apply combinational and sequential circuit elements to solve computational problems.
- Describe the concept of states and the sequential behaviour and control of digital circuits.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; LE/EECS 2021 4.00; LE/EECS2200 3.00

EECS 3213 3.00 Communication Networks

This course is an introduction to communications and networking. Topics covered include the distinction between information and data, between signal and data, between symbol and data, and between analogue and digital data; fundamental limits due to Shannon and Nyquist; protocol hierarchies; the OSI model; encoding of analogue/digital data; data link protocols; error and flow control; medium access; Ethernet and token passing systems in LANs; routing of packets in networks; transport services and protocols; high-level applications and their protocols.

Learning outcomes for the course:

- Explain network architectures in terms of the open systems interconnect (OSI) model and the role of each layer in the model.
- Classify networks in terms of topology, channelization, and routing characteristics.

- Explain and calculate physical communication limits imposed by wired and wireless channels in terms of delay, noise, and capacity.
- Explain and quantify means of signal representation for digital communication and describe communication system arrangements for forward and backward error control.
- Describe and analyse channelization strategies and medium access control techniques.
- Explain routing strategies through switched networks.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; SC/MATH1310 3.00

EECS 3214 3.00 Computer Network Protocols and Applications

This course focuses on the higher-level network protocols, security issues, network programming, and applications. Topics covered include networking basics; queuing fundamentals; network layer protocols including ICMP, DHCP and ARP multicasting; transport layer UCP and TCP, sockets and socket programming; application layer protocols including HTTP and DNS; multimedia; security; VOIP.

Learning outcomes for the course:

- Describe the different parts of the TCP/IP architecture, including their functionality, strengths and weaknesses, and the design principles used in their construction
- Describe several infrastructural applications such as FTP, HTTP and DNS, and third-party applications like P2P systems
- Write small applications using socket programming
- Contrast the service models and features of transport layer protocols (TCP and UDP), and describe the design of TCP congestion control algorithms
- Discuss issues affecting the choice of routing algorithms in practice, based on a deep understanding of the network layer
- Determine the security needs of different parts of the TCP/IP architecture and evaluate the existing security features based on knowledge of public- and private-key cryptosystems
- Discuss how to alleviate problems in multimedia transmission over the Internet, and evaluate existing multimedia protocols like SIP and RTCP

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00
NCR Note. This course is not open for credit to students who passed SC/CSE4213 3.00

EECS 3215 4.00 Embedded Systems
Introduction to the design of embedded systems using both hardware and software. Topics include microcontrollers; their architecture and programming; design and implementation of embedded systems using field programmable gate arrays. Topics covered include introduction to a specific microcontroller architecture, its assembly language, and programming; peripherals, input/output ports and timers; interrupts; memory systems; analog to digital and digital to analog conversion; parallel and serial interfacing; hardware modelling; structural specification of hardware; rapid prototyping using FPGA's.

Learning outcomes for the course:

- Select and utilize appropriate parallel, serial and analog interfaces.
- Design embedded software and hardware systems to address problems in important application domains under tight constraints.
- Design, implement and interface with standard and custom peripherals.
- Prototype embedded systems using microcontrollers and field programmable gate arrays (FPGAs).
- Understand embedded microcontroller architecture, development, debugging and testing.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; LE/EECS2031 3.00; LE/EECS3201 4.00

EECS 3221 3.00 Operating System Fundamentals
This course is intended to teach students the fundamental concepts that underlie operating systems, including multiprogramming, concurrent processes, CPU scheduling, deadlocks, memory management, file systems, protection and security. Many examples from real systems are given to illustrate the application of particular concepts. At the end of this course, a student will know the principles and techniques required for understanding and designing operating systems.

Learning outcomes for the course:

- Explain the fundamental concepts that underlie operating systems, including multiprogramming, concurrent processes, CPU scheduling, deadlocks, memory management, file systems, protection and security.
- Explain algorithms, structures, and mechanisms that are used in operating systems.
- Analyse the performance of process management methods and memory management schemes in operating systems.
- Design and implement single programs using concurrent processes and threads.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; LE/EECS2021 4.00; LE/EECS2031 3.00

EECS 3301 3.00 Programming Language Fundamentals (not offered in 2018/2019)

The topic of programming languages is an important and rapidly changing area of computer science. This course introduces students to the basic concepts and terminology used to describe programming languages. Instead of studying particular programming languages, the course focuses on the "linguistics" of programming languages, that is, on the common, unifying themes that are relevant to programming languages in general. The algorithmic or procedural, programming languages are particularly emphasised. Examples are drawn from early and contemporary programming languages, including FORTRAN, Algol 60, PL/I, Algol 68, Pascal, C, C++, Eiffel, Ada 95, and Java.

This course is not designed to teach any particular programming language. However, any student who completes this course should be able to learn any new programming language with relative ease.

Prerequisites: General prerequisites; LE/EECS2011 3.00; LE/EECS2001 3.00

EECS 3311 3.00 Software Design

A study of design methods and their use in the correct construction, implementation, and maintenance of software systems. Topics include design, implementation, testing, documentation needs and standards, support tools. Students design and implement components of a software system.

This course focuses on design techniques for both small and large software systems. Techniques for the design of components (e.g., modules, classes, procedures, and executables) as well as complex architectures will be considered. Principles for software design and rules for helping to ensure software quality will be discussed. The techniques will be applied in a set of small assignments, and a large-scale project, where students will design, implement, and maintain a non-trivial software system.

Learning outcomes for the course:

- Describe software specifications via Design by Contract, including the use of preconditions, postconditions, class invariants, loop variants and invariants.
- Implement specifications with designs that are correct, efficient and maintainable.
- Develop systematic approaches to organizing, writing, testing and debugging software.
- Develop insight into the process of moving from an ambiguous problem statement to a well-designed solution.
- Design software using appropriate abstractions, modularity, information hiding, and design patterns.
- Develop facility in the use of an IDE for editing, organizing, writing, debugging, testing and documenting code including the use of BON/UML diagrams for documenting designs. Also the ability to deploy the software in an executable form.
- Write precise and concise software documentation that also describes the design decisions and why they were made.

Prerequisites: General prerequisites; LE/EECS2030 3.00 or LE/EECS 1030; SC/MATH1090 3.00

EECS 3341 3.00 Introduction to Program Verification (not offered in 2018/2019)

Application of logic to programs; weakest precondition; semantics of a simple programming language; correctness; development of correctness proofs from specifications; application to software design; performance bounds; transformation and synthesis.

Prerequisites: General prerequisites; LE/EECS2011 3.00; LE/EECS2031 3.00; SC/MATH1090 3.00

Course Credit Exclusions: AK/AS/SC/COSC 3111 3.00

EECS 3342 3.00 System Specification and Refinement
This course provides students with an understanding of how to use mathematics (set theory and predicate logic) to specify and design correct computer systems whether the systems are sequential, concurrent or embedded. The course stresses both the underlying theory as well as the ability to use industrial strength tools that can be applied in practice. User requirements are formalized via an abstract mathematical model that is amenable to formal reasoning long before any programming activity is undertaken (e.g. as done in Event-B, Z and VDM). Successive models are like blueprints in traditional engineering disciplines and their mathematical nature allows us to reason about and predict their safety properties.

Learning outcomes for the course:

- Document requirements organizing them into appropriate categories such as environmental constraints versus functional properties (safety and progress).
- Construct high level, abstract mathematical models of a system (consisting of both the system and its environment) amenable to formal reasoning.
- Apply set theory and predicate logic to express functional and safety properties from the requirements as events, guards, system variants and invariants of a state-event model.
- Use models to reason about and predict their safety and progress properties.
- Plan and construct a sequence of refinements from abstract high-level specifications to implemented code.
- Prove that a concrete system refines an abstract model.
- Apply the method to a variety of systems such as sequential, concurrent and embedded systems.
- Use practical tools for constructing and reasoning about the models.
- Use Hoare Logic and Dijkstra weakest precondition calculus to derive correct designs.

Prerequisites: General prerequisites; LE/EECS2011 3.00; SC/MATH 1090 3.00

EECS 3401 3.00 Introduction to Artificial Intelligence and Logic Programming
Artificial Intelligence (AI) deals with how to build systems that can operate in an intelligent fashion. In this course, we examine fundamental concepts in AI: knowledge representation and reasoning, search, constraint satisfaction, reasoning under uncertainty, etc. The course also introduces logic programming, a programming paradigm based on predicate logic, where one specifies problems in a declarative way and one can use the language to search for a solution. Students will learn how to develop programs in Prolog to solve AI problems. Topics covered include introduction to AI and intelligent agents; logical representations, first-order syntax and semantics; basics of logic programming and Prolog; inference in first order logic; reasoning with Horn theories; search; constraint satisfaction; uncertain reasoning, Bayes Nets.

Learning outcomes for the course:
- Define the main objectives of artificial intelligence.
- Describe how first-order predicate logic forms the basis of logic programming.
- Write logic programs in Prolog.
- Use and modify heuristic state-space search algorithms such as A*, RTA* and IDA*.
- Represent knowledge in a small domain using predicate logic and use the representation to build a logical database for a knowledge-based expert system.
- Describe and use some other AI techniques including backward and forward chaining, Bayesian networks, game search and constraint satisfaction, grammars for natural language processing.

Prerequisites: General prerequisites; LE/EECS2011 3.00; SC/MATH1090 3.00

EECS 3403 3.00 Platform Computing (not offered in 2018/2019)
This course presents the .NET platform and in all topics, as applicable, compares this platform to JEE and other platforms such as Mono, Ruby on Rails, Django, etc. Also, the course discusses how platform computing has affected and affects major web paradigms, such as the traditional World Wide Web, Web 2.0, Semantic Web/Web 3.0, and W4 (World Wide Wisdom Web).

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00,

EECS 3421 3.00 Introduction to Database Systems

Concepts, approaches and techniques in database management systems (DBMS) are taught. Topics include logical models of relational databases, relational database design, query languages, crash recovery, and concurrency control.

The purpose of this course is to introduce the fundamental concepts of database management, including aspects of data models, database languages, and database design. At the end of this course, a student will be able to understand and apply the fundamental concepts required for the design and administration of database management systems. Topics covered include overview of database management systems; relational model; entity-relational model and database design; SQL; integrity constraints; crash recovery; concurrency control.

Learning outcomes for the course:

- Model databases proficiently at conceptual and logical levels of design. Use entity relationships (ER) models and ER diagrams with extension.
- Develop relational database schemas which respect and enforce data integrity represented in ER models.
- Implement a relational database schema using structured query language (SQL): create and manipulate tables, indexes, and views.
- Create and use complex queries in SQL.
- Write database application programs with an understanding of transaction management, concurrency control, and crash recovery.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00
Course Credit Exclusions: AP/ITEC3220 3.00

EECS 3431 3.00 Introduction to 3D Computer Graphics

This course introduces the fundamental concepts and algorithms of three-dimensional computer graphics. Topics include: an overview of graphics hardware, graphics systems and APIs, object modelling, transformations, camera models and viewing, visibility, illumination and reflectance models, texture mapping and an introduction to advanced rendering techniques such as ray tracing. Optional topics include an introduction to animation, visualization, or real-time rendering.

Learning outcomes for the course:

- Explain the basic stages and concepts of a modern graphics pipeline.
- Model a virtual scene using geometric primitives and affine transformations.
- Use mathematical formulas to animate elements of a virtual scene.
- Model basic materials and their interaction with virtual light sources.
- Explain basic concepts related to colour spaces and visual perception.
- Explain basic concepts related to global illumination.
- Produce rendered images of virtual scene from a corresponding scene description file.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; SC/MATH1025 3.00
Course credit exclusions: AK/AS/SC/CSE4431 3.00

EECS 3451 4.00 Signals and Systems

The study of computer vision, graphics and robotics requires background in the concept of discrete signals, filtering, and elementary linear systems theory. Discrete signals are obtained by sampling continuous signals. Starting with a continuous time signal, students will review the concept of a discrete signal, the conditions under which a continuous signal is completely represented by its discrete version, and discuss the analysis and design of linear time-invariant systems. In particular, frequency selective filters in both discrete and continuous time domain will be developed. An accompanying covers applications of the concepts covered in the lectures to practical problems such as speech and image processing. There are three supervised lab hours per week.

Topics covered include continuous and discrete time signals, linear time-invariant systems; Fourier analysis is continuous and discrete time; sampling; Laplace transform; Z transform; linear feedback systems; design of continuous and discrete time frequency selective filters.

Learning outcomes for the course:

- Explain how continuous and discrete-time signals can be represented in both time and frequency domains.

- Represent linear systems both as systems of differential/difference equations and in terms of frequency response.
- Describe and use the principles of linear time invariant systems and the properties of Fourier and Laplace transforms.
- Analyse the effects of discrete-time representation of continuous signals.
- Design, build and measure continuous and discrete time frequency selective filters.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; LE/EECS 2021 4.00; SC/MATH 1310 3.00
Course Credit Exclusions: LE/ESSE4020 3.00, SC/EATS4020 3.00, SC/MATH4130B 3.00, SC/MATH4830 3.00, SC/PHYS4060 3.00.

EECS 3461 3.00 User Interfaces
This course introduces the concepts and technology necessary to design, manage and implement user interfaces (UIs). Users are increasingly more critical towards poorly designed interfaces. Consequently, for almost all applications more than half of the development effort is spent on the UI.
The first part of the course concentrates on the technical aspects of UIs. Students learn about event-driven programming, windowing systems, widgets, the model-view-controller concept, UI paradigms, and input/output devices.
The second part discusses how to design and test UIs. Topics include basic principles of UI design, design guidelines, UI design notations, UI evaluation techniques, and user test methodologies
The third part covers application areas such as groupware (CSCW), multi-modal input, UIs for Virtual Reality, and UIs for the WWW.
Students work in small groups and utilise modern toolkits and scripting languages to implement UIs. One of the assignments focuses on UI evaluation.

Learning outcomes for the course:
- Explain the capabilities of both humans and computers from the viewpoint of human information processing.
- Describe and critically evaluate typical human–computer interaction (HCI) models, styles, and various historic HCI paradigms.
- Apply an interactive design process and universal design principles to designing HCI systems.

- Describe and apply HCI design principles, standards and guidelines.
- Analyse, identify and critically evaluate user models, user support, socio- organizational issues, and stakeholder requirements of HCI systems.
- Analyse, discuss and critically evaluate HCI issues in groupware, ubiquitous computing, virtual reality, multimedia, and Word Wide Web related environments.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00
Course Credit Exclusions: AP/ITEC3230 3.00, ITEC3461 3.00

EECS 3481 3.00 Applied Cryptography
This course provides an overview of cryptographic primitives in the context of computer security and looks at how they are applied to protect communication patterns. The emphasis is on the applied aspects as used in software to protect applications and build secure protocols. Topics covered include Foundation: security goals, the communication model, classification of attacks; Classical Cryptography: classical ciphers, diffuse and confuse, information theory, and cryptanalysis; Modern Cryptography: modern symmetric ciphers, perfect secrecy, block and stream ciphers, modern asymmetric ciphers; Hash Functions: message integrity, digital signatures, certificate authorities, key distribution protocols; Advanced topics and applications may include secret sharing, zero-knowledge proofs, quantum cryptography, and digital cash.

Learning outcomes for the course:

- Explain the workings of fundamental cryptographic algorithms in classical, symmetric, and asymmetric settings, and apply them programmatically.
- Attack a given communication pattern using exhaustive as well as cryptanalytic techniques such as meet-in-the-middle, person in the middle, or birthday, or by exploiting an algorithmic vulnerability.
- Analyse a given communication pattern with respect to achieving certain goals in a security context by identifying vulnerabilities, threats, and risks and recommending hardening mechanisms.
- Apply cryptographic primitives in advanced settings such as secret sharing, zero-knowledge, and digital cash.

- Explain the impact of advances in computing power, algorithm complexities, and quantum computing, on the strength of cryptographic algorithms.

Prerequisites: General prerequisites; LE/EECS2011 3.00

EECS 3482 3.00 Introduction to Computer Security
This course introduces students to the basic concepts, goals and terminology of computer security. It provides a general overview of the computer security body of knowledge with an emphasis on the risk-based mind-set that a computer security professional needs to have. Students will be exposed to both the theoretical and the practical aspects of computer security (the lab sessions will include security case studies as well as exercises using modern security tools).

Learning outcomes for the course:

- Describe different categories of modern-day computer security threats and attacks, and explain their impact.
- Differentiate between the main categories of modern day malware.
- Discuss the basics of classical and modern cryptography and steganography techniques, and apply those techniques and related software tools in applications involving data confidentiality.
- Describe the basics of project management theory and security risk management as they pertain to computer security in an organization.
- Apply qualitative and quantitative security risk analysis (including cost-benefit formulas) to real-world case studies.
- Discuss how criminal and civil law applies to computer-related offences and computer security, including the protection of intellectual property.

Three lecture hours per week. Two lab hours every other week.

Prerequisites: Any 12 university credits at the 2000-level in any discipline

EECS 3505 3.00 Electrical Systems for Mechanical Engineers
This is a course that develops the knowledge and skills mechanical engineers need to interact with electrical/electronic systems. Topics covered include Electronic Components-Devices, Electrical Noise, Introduction to Electro-magnetism, DC Machines,Transformer & Three-

Phase AC Circuit, Three-Phase Induction Motors, Synchronous Machines, Special Machines include stepper, linear and BLDC motors, and communication protocols.

Learning outcomes for the course:

- Comprehend the basic concepts of circuits as well as electronic components and apply simulation tools to model their inputs and outputs relationships
- Explain the characteristics of electronic devices such as diode, operational amplifier, laches, etc, and evaluate the behaviour of both passive and active filters including low pass, high pass and band pass.
- Design and employ noise reduction strategies to improve signal quality.
- Recognize the two basic principles (i.e., generation of force and EMF) related to electromechanical energy conversion.
- Identify the principles of operation of different machine types including: DC machines, induction machines, synchronous machines, stepper motors, linear motors and servo motors.

Prerequisites: SC/PHYS 1801 3.0; LE/MECH 2502 3.0

EECS 3602 4.00 Systems and Random Processes in Discrete Time
Discrete time signals are obtained by sampling continuous signals. Starting with a continuous time signal, the course reviews the concept of a discrete time (DT) signal, the conditions under which a continuous signal is completely represented by its discrete version, and discusses the analysis and design of linear time- invariant, discrete-time systems. In particular, frequency selective filters in the discrete time domain are developed. The second half of the course will cover advanced topics of random processes, noise, and their applications in the real world including the effect of linear systems on the statistical properties of random signals. The course includes a lab that applies the theoretical concepts and algorithms learned in the course to practical, real-world applications. Topics covered in the course include Introduction to DT Signals and Systems; Properties of DT Systems; Representations for Linear, Time Invariant DT Systems: Difference Equations; Convolution Sum; Z/Fourier Transfer functions; Design of DT Filters: FIR filters and IIR Filters; digital communication

systems and information theory; Random variables in one and multiple dimensions: Expectation, higher order moments, probability density functions, transformations, moment generating functions, and characteristic functions; Random Sequences and Processes; Types of random processes (WSS, SSS, and Ergodic processes); Output of LTI Systems with random inputs, Noise, and Wiener-Khinchin Theorem.

Learning outcomes for the course:

- Represent a continuous-time signal as a DT signal without any information loss.
- Compute the steady state outputs of linear time-invariant systems in the continuous-time domain using three different but equivalent techniques:
    1. Solving difference equations,
    2. Convolution with the impulse response, and;
    3. The Fourier (or, alternatively, the Z) transform.
- Calculate the frequency representations (Fourier and Z-transforms) of periodic and aperiodic DT signals.
- Represent a DT linear time invariant system using its magnitude and phase spectrum.

Prerequisite: General prerequisites, SC/MATH 2930; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; LE/EECS 2602 4.00
Course Credit Exclusion: LE/EECS3451 4.00

EECS 3603 4.00 Electromechanical Energy Conversion
This is an introductory course for energy conversion. It covers the basic construction, principle of operations and solid-state control of different types of electric machines including both AC and DC machines. In particular, the following list of electric machines will be covered: DC machines, single-phase and poly-phase transformers, three-phase induction motors, synchronous machines and special machines (stepper, linear and servo motors).

Laboratory exercises investigate topics in electromagnetism, DC machines, transformers, induction machines, synchronous machines, and special machines.

Learning outcomes for the course:

- Understand the basic concepts of magnetic circuits as applied to electric machines.
- Understand the two basic principles (generation of force and emf) that govern electromechanical energy conversion.
- Describe fundamental principles of energy conversion, which are the analytical foundations for understanding all types of drives.
- Identify the principles of operation of different machine types including: DC machines, transformers, induction machines, synchronous machines, stepper motors, linear motors and servo motors.
- Derive the steady-state modelling (equivalent circuits) of different types of machines.
- Analyse the steady-state performance and input-output operational characteristics of the different types of machines.
- Learn to use space vectors presented on a physical basis to describe the operation of an Ac machine.
- Describe solid-state control strategies for the different machine types.
- Present some motor and generator application.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; LE/EECS2200 3.00; SC/PHYS2020 3.00

EECS 3604 4.00 Electromagnetic theory and wave propagation
The objective of this course is to provide the student with: an introduction to partial differential equations; the mathematics of wave propagation; and the science of electromagnetic wave propagation. Beginning with a review of vector calculus, the course introduces solutions to partial differential equations. Maxwell's equations are presented as a motivating example for partial differential equations, and the EM wave equation is derived from Maxwell's equations in multiple dimensions. Solutions for propagation in waveguides and transmission lines are given, and antennas are introduced. Students are also introduced to solutions of partial differential equations beyond EM, such as the diffusion equation.

Learning outcomes for the course:

- Comprehends the meaning of a partial differential equation (PDE) in a mathematical expression.

- Recognizes the wave equation as a type of PDE in a mathematical expression;
- Applies the travelling wave as a solution to the wave equation in a mathematical expression, and produces this solution numerically in the laboratory.
- Comprehends Maxwell's equations as physical phenomena, in mathematical expressions and in written descriptions, and identifies the effects of Maxwell's equations in the laboratory.
- Computes the wave equation from Maxwell's equations in one, two, and three dimensions, and applies the travelling wave solution, in a mathematical expression.
- Computes the travelling wave solution in a conducting waveguide and in a transmission line, and applies these solutions in the laboratory.
- Analyses the operation of EM antennas in mathematical expressions, and in the laboratory.
- Differentiates between the wave equation and the diffusion equation in a mathematical expression.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; SC/MATH1014 3.00; SC/MATH1025 3.00; SC/PHYS2020 3.00

EECS 3610 4.00 Semiconductor Physics and Devices

The course first introduces the students to semiconductor physics such as carrier statistics, band diagrams and different conduction mechanisms. This knowledge is then applied to the most important semiconductor devices, namely diodes and metal-oxide-semiconductor field- effect transistors (MOSFETs). This includes understanding of the basic elements that make up these devices: p-n-junctions, metal-semiconductor junctions and metal-oxide-semiconductor capacitors (MOSCAPs). Other field-effect transistors (junction field-effect transistor (JFET), thin-film transistor (TFT)) are introduced and contrasted with MOSFETs. The course also includes a brief discussion of semiconductor manufacturing. The course concludes with state-of-the-art MOSFET scaling challenges and solutions such as high-k dielectrics, strain and FinFET in the context of the International Technology Roadmap for Semiconductors (ITRS) and its successor the International Roadmap for Devices and Systems (IRDS). Students use device simulation software to model device behavior and gain device simulation skills.

Learning outcomes for the course:

- Understand the physical mechanisms that underlie the behavior of carriers inside semiconductors.
- Demonstrate the ability to analyze different semiconductor devices in terms of characteristics such as current-voltage, capacitance-voltage.
- Design devices by varying device parameters such as doping levels, geometry or material to create desired performance characteristics.
- Design basic microfabrication process flows.
- Be able to apply novel device designs to solve issues resulting from the continued need for scaling.
- Demonstrate the ability to perform device simulations and device design employing TCAD software.

Prerequisite: General prerequisites; LE/EECS 2210 3.00 or SC/PHYS 3050 3.00

EECS 3611 4.00 Analog Integrated Circuit Design

The course focuses on the analysis and design of analog integrated circuits in CMOS technology. It covers basic MOS device physics, basic circuit models for single-stage amplifiers, differential amplifiers, current mirrors, frequency response of amplifiers, operational amplifiers, and layout techniques. The course includes a mandatory lab that introduces the computer-aided design software to performance circuit simulation and evaluation. Topics to be covered includes introduction to analog design; Basic MOS device physics; Single---stage amplifiers; Differential amplifiers; Passive and active current mirrors; Frequency response of amplifiers; Noise; Feedback; Operational amplifiers; Layout and design rules.

Learning outcomes for the course:

- To analyse the characteristics of basic analog integrated circuits.
- To formulate the behaviour of basic analog circuits by inspection.
- To perform circuit simulation using computer-aided tool.
- To draw layout based on given design rules.

Prerequisite: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; LE/EECS2210 3.00

EECS 3612 4.00 Introduction to Sensors and Measurement Instruments

The course covers the theory, principle and technology of sensors and transducers by developing the students' understanding of fundamental of measurement instruments encompassing sensors, interface circuits to transfer the measured data for monitoring and/or further signal processing purposes. Background in electronic circuit design, basic knowledge of physics, chemistry and molecular/cellular biology are required before taking the course. There will be small design projects for the labs to reinforce sensor interfacing. In the design of each instruments, students are also introduced the related theoretical and practical issues with a focus on needs assessment, creativity, and innovation as they seek to identify market opportunities. Topics include the fundamental of physical, chemical, and biological sensors used for various applications including health or industrial applications; design of interface circuit dedicated to each sensor; quality factors of sensors' performance including resolution, dynamic range and sensitivity. Also this course enables the electrical engineering students to learn the principle and operation of main electrical measurement instruments. The state of art sensor technology will also be discussed briefly. A detailed list of topics covered within this course is as follows.

Learning outcomes for the course:

The primary objective will be on developing the student's ability to integrate and apply their knowledge in electronic circuits and classic knowledge in physical, chemical and biological science in the design of measurement instruments using advanced sensing technologies. Specifically, students will:

- Investigate measurement-sensing techniques used for various applications.
- Use electrical instruments and tools for the measurement of voltage, current, power and energy.
- Design interface circuits and systems to record sensing data into a computer for monitoring and simple signal processing purposes.

- Analyse the measurement question and select an appropriate the sensor technology and interfacing method to develop the instrument.
- Describe the state of art sensing technology and the market challenges in this field.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; LE/EECS2210 3.00

EECS 3900 0.00 Computer Science Internship Work Term

This experiential education course reflects the work term component of the internship option. Qualified Honours students gain relevant work experience as an integrated complement to their academic studies, reflected in the requirements of a learning agreement and work term report. Students are required to register in this course for each four month work term, with the maximum number of work term courses being four (i.e., 16 months). Students in this course receive assistance from the Lassonde internship office prior to and during their internship, and are also assigned a Faculty Supervisor/Committee.

Prerequisites:
Enrolment is by permission only. Criteria for permission include: 1. That students have successfully completed at least 9.00 EECS credits at the 3000-level including EECS 3311 3.00, with a Grade Point Average (GPA) of at least 6.00 in all mathematics and computer science courses completed; 2. That students are enrolled full-time in the Honours program prior to beginning their internship and have attended the mandatory preparatory sessions as outlined by the Lassonde internship office; 3. That students have not been absent for more than two consecutive years as a full-time student from their Honours degree studies; 4. That upon enrolling in this course, students have a minimum of 9 credits remaining toward their Honours degree and need to return as a full-time student for at least one academic term to complete their degree after completion of their final work term.

Note: This is a pass/fail course, which does not count for degree credit. Registration in EECS 3900 0.00 provides a record on the transcript for each work term.

EECS 3980 0.00 Computer Security Internship Work Term

This experiential education course reflects the work term component of the internship option. Qualified Honours students gain relevant work experience as an integrated complement to their academic studies, reflected in the requirements of a learning agreement and work term report. Students are required to register in this course for each four month work term, with the maximum number of work term courses being four (i.e., 16 months). Students in this course receive assistance from the Lassonde internship office prior to and during their internship, and are also assigned a Faculty Supervisor/Committee.

Prerequisites: Enrolment is by permission only. Criteria for permission include: 1. That students have successfully completed at least 9.00 EECS credits at the 3000-level including EECS 3482 3.00, with a Grade Point Average (GPA) of at least 6.00 in all mathematics and computer science courses completed; 2. That students are enrolled full-time in the Honours program prior to beginning their internship and have attended the mandatory preparatory sessions as outlined by the Lassonde internship office; 3. That students have not been absent for more than two consecutive years as a full-time student from their Honours degree studies; 4. That upon enrolling in this course students have a minimum of 9 credits remaining toward their Honours degree and need to return as a full-time student for at least one academic term to complete their degree after completion of their final work term.

Note: This is a pass/fail course, which does not count for degree credit. Registration in EECS 3980 0.00 provides a record on the transcript for each work term.

Course Descriptions: 4000-Level

EECS 4070 3.00 Directed Studies

This is a course for advanced students who wish to carry out independent study on a topic within EECS that is not offered in a regular course during a particular academic session. The student must identify a faculty member with expertise in the area that is willing to supervise the student's work.

At the beginning of the term, the student and faculty supervisor must prepare a written description of the course, its content, and the method of evaluation. The work involved must be equivalent to a three-credit course at the 4000 level, and the course coordinator must confirm this before the student is permitted to enrol. The evaluation will generally be based on written work and the student's ability to discuss the course material during meetings with the supervisor. In addition, assigned work could include oral or written presentations of material for non-specialists.

The course coordinator, student and supervising faculty member should each retain a copy of the agreed-upon description of the course, and relevant details should be entered into the student's record via the student information system (SIS).

Learning outcomes for the course:

- Create a literature review.
- Gather in-depth knowledge of a topic independently.
- Communicate the results of independent learning orally and in writing.

Prerequisites: General prerequisites; successful completion of 24 credits in LE/EECS major courses *and* permission of course coordinator

EECS 4080 3.00 Computer Science Project

This is a course for advanced students, normally those in the fourth year of an honours program, or students who have passed 36 computer science credits. Students who have a project they wish to do need to convince a member of the faculty in the Department that it is appropriate for course credit.

Alternatively, students may approach a faculty member in the Department (typically, one who is teaching or doing research in the area of the project) and ask for project suggestions. Whatever the origin of the project, a "contract" is required. It must state the scope of the project, the schedule of work, the resources required, and the criteria for evaluation. The contract must be signed by the student and his/her project supervisor and be acceptable to the course director. *A critical course component that must be included in the contract is a formal seminar presentation.* The course director will arrange the seminar sessions, and students and their faculty supervisors are required to participate. The seminar talks will have a typical length of 15-20 minutes, and will be evaluated by the individual supervisor, the course director and one more faculty member. This talk will be worth 30% of the final mark. The remaining 70% of the course mark is the responsibility of the individual supervisor.

Prerequisites: General prerequisites and permission of the course director. Restricted to students who have passed 36 credits in Computer Science.
Course Credit Exclusions: LE/EECS 4081 6.00, LE/EECS 4082 6.00, LE/EECS 4084 6.00, LE/EECS4088 6.00, LE/EECS 4480 3.00, ENG4000 6.00

EECS 4088 6.00 Computer Science Capstone Project

This is a course for students in the fourth year of an honours program. Students who have a project they wish to do need to convince a member of the faculty in the Department that it is appropriate for course credit. Alternatively, students may approach a faculty member in the Department (typically, one who is teaching or doing research in the area of the project) and ask for project suggestions. Whatever the origin of the project, a “contract” is required. It must state the scope of the project, the schedule of work, the resources required, and the criteria for evaluation. The contract must be signed by the student and his/her project supervisor and be acceptable to the course director. A critical course component that must be included in the contract is a final presentation. The course director will arrange the final presentation session, and students and their faculty supervisors are required to participate. The presentations will be evaluated by the individual supervisor, the course director and one more faculty member. This presentation will be worth 30% of the final mark. The remaining 70% of the course mark is the responsibility of the individual supervisor.

Prerequisites: General prerequisites and permission of the course director. Normally restricted to students who have passed 36 credits in Computer Science.
Course Credit Exclusions: LE/EECS4080 3.00, LE/EECS4081 6.00, LE/EECS4082 6.00, LE/EECS4084 6.00, LE/EECS4480 3.00, ENG4000 6.00

EECS 4090 6.00 Software Development Capstone Project

A well-designed software product is more than just a computer program. A software product consists of quality code, a well thought out design developed via disciplined professional engineering standards, appropriate literate documentation including requirements, design and testing documents, a manual, and the appropriate installation files and instructions needed to get the product to work. The product has to be correct (i.e., it must satisfy all the requirements specified by the client), usable, efficient, safe and maintainable.

This course provides students with an opportunity to integrate what they have learned in earlier computer science courses, deepen their understanding of that material, extend their area of knowledge, and apply their knowledge and skills in a realistic simulation of professional experience. The end result must be a substantial software product.

This course is run on a tight schedule over the Fall and Winter Terms; work is on-going and regular. The course is intended to help with the transition from being a student to being an active professional in industry. During the course students are expected to perform independent study, plan their work, make decisions, and take ownership of the consequences of their mistakes.

A combination of teamwork and individual work is required. The requirements elicitation, requirements analysis, design, coding, testing, and implementation of the product will be a team effort. However, individual responsibilities must be clearly identified in every deliverable.

This project will be of significant size and like most industrial projects it will be time and resource limited. Students must meet the specified deadlines. As a result, they will have to set their goals and plan their work accordingly.

Students must apply sound mathematics, good engineering design, and algorithms throughout the project. However, they will also need to apply heuristics and design patterns, or “rules of thumb”, where sound, well-understood algorithms are not available. Any such heuristics must be clearly identified and supported by arguments that justify their choice. The teams will be required to show that the heuristic cannot fail in a way that will violate safety restrictions or other restrictions designated as critical.

Learning outcomes for the course:

- Describe the requirements of a large software system.

- Select appropriate system elements for a high-level design description.
- Derive and implement test cases at the unit and system level.
- Produce a detailed user manual for an interactive system.
- Implement a large software system from scratch.

Prerequisites: Only open to students in the Software Development Stream. LE/EECS3311 3.00 with a grade of B or higher; LE/EECS 3101 3.00; LE/EECS 3342 3.00
Co requisites: LE/EECS 4312 3.00, LE/EECS 4313 3.00

EECS 4101 3.00 Advanced Data Structures (integrated with EECS5101 3.00)

The course discusses advanced data structures such as heaps, balanced binary search trees, hashing tables, red-black trees, B-trees and their variants, structures for disjoint sets, binomial heaps, Fibonacci heaps, finger trees, persistent data structures, self-adjusting data structures, etc. When feasible, a mathematical analysis of these structures will be presented, with an emphasis on average case analysis and amortised analysis. If time permits, some lower bound techniques may be discussed, as well as NP-completeness proof techniques and approximation algorithms.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; LE/EECS2001 3.00; LE/EECS3101 3.00

EECS 4111 3.00 Automata and Computability (integrated with EECS5111 3.00)

This course is the second course in the theory of computing. It is intended to give students a detailed understanding of the basic concepts of abstract machine structure, information flow, computability, and complexity. The emphasis will be on appreciating the significance of these ideas and the formal techniques used to establish their properties. Topics chosen for study include: models of finite and infinite automata, the limits to computation, and the measurement of the intrinsic difficulty of computational problems.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; LE/EECS2001 3.00; LE/EECS3101 3.00

EECS 4115 3.00 Computational Complexity

This course provides an introduction to complexity theory, one of the most important and active areas of theoretical computer science. Students learn basic concepts of the field and develop their abilities to read and understand published research literature in the area and to apply the most important techniques in other areas. Topics covered include models of computation for complexity: Turing machines, random access machines, circuits and their resources such as time, space, size, and depth; time- and space-bounded diagonalisation, complexity hierarchies, resource bounded reducibility such as log space and polynomial time reducibility; P vs. NP: nondeterminism, Cook's Theorem and techniques for proving NP-Completeness; Nondeterministic space: The Savitch and Immerman/Szelepcsenyi Theorems; Important complexity classes (and natural problems complete for them) including: P, NP, co-NP, the Polynomial time Hierarchy, log space, Polynomial SPACE and Exponential time.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; LE/EECS2001 3.00; LE/EECS3101 3.00

EECS 4141 3.00 Introduction to Quantum Computing (not offered in 2018/2019)

This course presents an overview of the quantum computing field without assuming any prior exposure to quantum mechanics. Drawing parallels between quantum and classical computing, the course covers the physical layer briefly before moving to quantum gates, the circuit model, and quantum algorithms. Quantum information is covered through applications.

Learning outcomes for the course:

- Employ linear algebra to represent, transform, and measure a given multi-qubit state.
- Design a quantum circuit using single and multi-qubit gates or determine the function of a given circuit
- Analyze a given search or phase estimation quantum algorithm
- Demonstrate the potential of quantum computers and describe the challenges in building them
- Build a secure channel using quantum cryptogaphy

Prerequisites: General prerequisites; SC/MATH 1025 3.00; LE/EECS2021 4.00; LE/EECS3101 3.00

EECS 4161 3.00 Mathematics of Cryptography (Cross-listed with SC/MATH 4161 3.00)

Cryptography deals with the study of making and breaking secret codes.

In this course, we will be studying situations that are often framed as a game between three parties: a sender (e.g., an embassy), a receiver (the government office) and an opponent (a spy). We assume that the sender needs to get an urgent message to the receiver through communication channels that are vulnerable to the opponent. To do this communication, the sender and receiver agree in advance to use some sort of code, which is unlocked by a keyword or phrase. The opponent will be able to intercept the message. Is he/she able to unlock the message without knowing the key?

We will learn some probability theory, information theory and number theory to answer questions about how vulnerable the methods of sending secrets are. This has a great number of applications to Internet credit card transactions, wireless communication and electronic voting. We will start by learning some classical codes (used up through WWI) and analysing those. The last third of the course we will start to learn the methods that are used in modern cryptography.

Prerequisites: At least 12 credits from 2000-level (or higher) MATH courses (without second digit 5, or second digit 7), or LE/EECS3101 3.00, or permission of the instructor.

EECS 4201 3.00 Computer Architecture (integrated with EECS5501 3.00)

This course presents the core concepts of computer architecture and design ideas embodied in many machines, and emphasises a quantitative approach to cost/performance tradeoffs. This course concentrates on uniprocessor systems. A few machines are studied to illustrate how these concepts are implemented; how various tradeoffs that exist among design choices are treated; and how good designs make efficient use of technology. Future trends in computer architecture are also discussed. Topics covered may include fundamentals of computer design; performance and cost; instruction set design and measurements of use;

pipeline design techniques; memory-hierarchy design; Input-output subsystems;

Learning outcomes for the course:

- Design cache, memory hierarchy, and virtual memory using different techniques to improve cost/performance ratio.
- Demonstrate how dynamic scheduling and speculative execution can improve the system performance and explain how it is implemented in modern processors.
- Evaluate different design alternatives and make quantitative/qualitative argument for one design over the other.
- Identity the different types of parallelism (data, instruction, thread, transaction) for a given application.
- Compare and evaluate different techniques (such as multithreading, multicore, or vector) to improve CPU performance.

Prerequisites: General prerequisites; LE/EECS3201 4.00, LE/EECS3221 3.00

EECS 4210 3.00 Architecture and Hardware for Digital Signal Processing (not offered in 2018/2019)

The field of DSP is driven by two major forces, advances in DSP algorithms, and advances in VLSI technology that implements those algorithms. This course addresses the methodologies used to design custom or semi-custom VLSI circuits for DSP applications, and the use of microcontrollers and digital signal processors to implement DSP algorithms. It also presents some examples of advances in fast or low power design for DSP. Topics to be covered may include basic CMOS circuits: manufacturing process, area, delay, and power dissipation; Implementation of fundamental operations: Carry lookahead adders, carry select adders, carry, save adders, multipliers, array multipliers, Wallace tree multipliers, Booth array multipliers, dividers, array dividers; Array processor architectures: Mapping algorithms into array processors; High High level architectural transformation for mapping algorithms into hardware: pipelining, retiming, folding, unfolding; Mapping DSP algorithms (FIR, IIR, FFT, and DCT) into hardware; Implementing DSP algorithms using microcontrollers; DSP support in general-purpose processors; The effect of scaling and roundoff noise.

Learning outcomes for the course:

- Map a DSP algorithm to a graphical representation and determine its fundamental lower bound on the achievable iteration or sampling period.
- Use pipelining, retiming, and parallel processing to improve the performance of a DSP implementation.
- Use folding technique to reduce silicon area in a DSP implementation.
- Assess alternative architectures based on a given set of design specifications.
- Implement a DSP algorithm based on an optimized architecture.

Prerequisites: General prerequisites; LE/EECS3201 4.00; one of LE/EECS3451 3.00 or LE/EECS3602 4.00

EECS 4211 3.00 Performance Evaluation of Computer Systems (not offered in 2018/2019) (integrated with EECS5422 3.00)

Topics covered may include a review of Probability Theory —probability, conditional probability, total probability, random variables, moments, distributions (Bernoulli, Poisson, exponential, hyperexponential, etc.); Stochastic Processes—Markov chains and birth and death processes; Queuing Theory—M/M/1 Queuing system in detail; other forms of queuing systems including limited population and limited buffers; case study involving use of the queuing theory paradigm in performance evaluation and modelling of computer systems such as open networks of queues and closed queuing networks; Use of approximation techniques, simulations, measurements and parameter estimation.

Prerequisites: General prerequisites; SC/MATH2030 3.00, LE/EECS3213 3.00

EECS 4214 4.00 Digital Communications

Digital communications has become a key enabling technology in the realization of efficient multimedia systems, wireless and wired telephony, computer networks, digital subscriber loop technology and other communication and storage devices of the information age. The course provides an introduction to the theory of digital communications and its application to the real world. Emphasis will be placed on covering design and analysis techniques used in source and channel coding, modulation and demodulation, detection of signal in the presence of noise, error detection and correction, synchronization, and spread spectrum. An introduction to information theory and recent development in the area will also be covered.

Topics covered in the course will be chosen from a review of probability and random variables; introduction to stochastic processes and noise; introduction to information theory: Shannon's source coding and channel coding theorems; source coding: lossless coding (Huffman, arithmetic, and dictionary codes) versus lossy coding (predictive and transform coding); analog to digital conversion: sampling and quantization; baseband transmission; binary signal detection and matched filtering; intersymbol interference (ISI), channel capacity; digital bandpass modulation and demodulation schemes; error performance analysis of M-ary schemes; channel coding: linear block, cyclic, and convolutional codes; decoding techniques for convolutional codes, Viterbi algorithm; application of convolutional codes to compact disc (CD); synchronization techniques; spread spectrum modulation: direct sequence and frequency hopping.

Learning outcomes for the course:

- Express and manipulate random signals in terms of probabilities and statistical averages.
- Understand and quantify the performance of key pulse code modulation systems.
- Understand and apply the theory of optimum detectors and filters for pulse amplitude modulation systems.
- Explain the operation of sequence detection methods.
- Describe the theory and operation of bandpass modulation and detection systems.
- Design and build system components capable of enabling optimum digital communication.

Prerequisites: General prerequisites; LE/EECS3213 3.00; one of SC/MATH2030 3.00 or SC/MATH2930 3.00; one of LE/EECS3451 4.00 or LE/EECS3602 4.00 or LE/ESSE4020 3.00 or SC/MATH4830 3.00 or SC/PHYS4060 3.00 or SC/PHYS4250 3.00

EECS 4215 3.00 Mobile Communications (integrated with EECS5431 3.00)

Wireless mobile networks have undergone rapid growth in the past several years. The purpose of this course is to provide an overview of the latest developments and trends in wireless mobile communications, and to address the impact of wireless transmission and user mobility on the design and management of wireless mobile systems.

Topics covered may include an overview of wireless transmission; wireless local area networks: IEEE 802.11, Bluetooth; 2.5G/3G wireless technologies; mobile communication: registration, handoff support, roaming support, mobile IP, multicasting, security and privacy; routing protocols in mobile ad-hoc networks: destination-sequence distance vector routing (DSDV), dynamic source routing (DSR), ad-hoc on-demand distance vector routing (AODV), and a few others; TCP over wireless: performance in and modifications for wireless environment; wireless sensor networks: applications; routing; satellite systems: routing, localization, handover, global positioning systems (GPS); broadcast systems: digital audio/video broadcasting; applications to file systems, world wide web; Wireless Application Protocol and WAP 2.0; i-mode; SyncML; other issues such as wireless access technologies, quality of service support, location management in mobile environments, and impact of mobility on performance.

Learning outcomes for the course:

- Explain the operation and purpose of key components in wireless communication systems.
- Calculate the link budget of a wireless communicator.
- Analyse the characteristics and impact of indoor and outdoor wireless channels.
- Analyse the coverage and throughput of wireless networks.
- Explain the channelization and control techniques employed by cellular and LAN networks.

Prerequisites: General prerequisites; LE/EECS3213 3.00

EECS 4221 3.00 Operating System Design (integrated with CSE5421 3.00)
An operating system has four major components: process management, input/output, memory management, and the file system. This project-oriented course puts operating system principles into action. This course presents a practical approach to studying implementation aspects of operating systems. A series of projects is included, making it possible for students to acquire direct experience in the design and construction of

operating system components. A student in this course must design and implement some components of an operating system and have each interact correctly with existing system software. The programming environment is C++ under Unix. At the end of this course, a student will be able to design and implement the basic components of operating systems.

Prerequisites: General prerequisites; LE/EECS3221 3.00
Course Credit Exclusions: COSC4321 3.00

EECS 4301 3.00 Programming Language Design (not offered in 2018/2019) (integrated with CSE5423 3.00)

This course is a continuation of EECS3301 Programming Language Fundamentals. Like its predecessor, the course focuses on the linguistics of programming languages; that is, on the common, unifying themes that are relevant to programming languages in general. Both algorithmic and non-algorithmic language categories are examined. Current techniques for the formal specification of the syntax and semantics of programming languages are studied. Skills are developed in the critical and comparative evaluation of programming languages.

Prerequisites: General prerequisites; LE/EECS3301 3.00

EECS 4302 3.00 Compilers and Interpreters (not offered in 2018/2019) (integrated with CSE5424 3.00)

Principles and design techniques for compilers and interpreters. Compiler organization, compiler writing tools, scanning, parsing, semantic analysis, run-time storage organization, memory management, code generation, and optimization. Students will implement a substantial portion of a compiler in a project.

This course is a hands-on introduction to the design and construction of compilers and interpreters. At the end of the course, you will understand the architecture of compilers and interpreters, their major components, how the components interact, and the algorithms and tools that can be used to construct the components. You will implement several components of a compiler or interpreter, and you will integrate these components to produce a working compiler or interpreter.

Prerequisites: General prerequisites; LE/EECS2011 3.00; LE/EECS3301 3.00 recommended

EECS 4311 3.00 System Development (not offered in 2018/2019)

System Development deals with the construction of systems of interacting processes. The course focuses on abstraction, specification, and analysis in software system development. Abstraction and specification can greatly enhance the understandability, reliability and maintainability of a system. Analysis of concurrency and interaction is essential to the design of a complex system of interacting processes.

The course is split into three parts. The first part discusses a semiformal method, Jackson System Development (JSD) by Michael Jackson. JSD is used to build an understanding of what system development entails and to develop a basic method of constructing practical systems of interacting processes. JSD gives precise and useful guidelines for developing a system and is compatible with the object-oriented paradigm.

The second part of the course discusses the mathematical model CSP (Communicating Sequential Processes by C.A.R. Hoare). While CSP is not suitable to the actual design and development of a system of interacting processes, it can mathematically capture much of JSD. Consequently, it is possible to use formal methods in analysing inter-process communication arising out of JSD designs.

The third part of the course discusses Z notation and its use in the specification of software systems. Z has been successfully used in many software companies — such as IBM and Texas Instruments — to specify and verify the correctness of real systems.

Prerequisites: General prerequisites; one of LE/EECS3311 3.00 or LE/EECS3221 3.00

EECS 4312 3.00 Software Engineering Requirements

Software requirement engineers are experts at eliciting the needs of their customers, translating customer needs into a precise requirements document (that describes what – not how – customer needs shall be satisfied), and providing systematic evidence-based methods to validate the requirements and verify that the final software product satisfies the requirements. Precise software requirements documents are especially needed in safety-critical cyber-physical systems (e.g., nuclear reactors, medical devices and transportation systems) and mission critical business systems (e.g., banking systems, health provision and cloud systems). A project will allow students to apply their knowledge to a small but non-trivial example.

Learning outcomes for the course:

- Elicit customer requirements by analyzing customer goals and needs
- Write precise requirements documents. To do this students will be able to:
- Develop the system overview, identify the system boundary and draw a context diagram
- Identify monitored variables and events and controlled variables
- Identify environmental assumptions and constraints
- Describe the functional requirements using tabular expressions (function tables) that specify the mathematical relation between the monitored variables and events and the controlled variables
- Describe the non-functional requirements
- Prove that the functional requirements are complete, disjoint and well-defined
- Provide a complete set of use cases and corresponding acceptance tests so that each requirement is a verifiable contract of customer needs
- Validate the functional requirements by proving that they preserve safety properties (derived from such methods as hazards analysis) and prove that the use cases satisfy the function tables

Prerequisites: General prerequisites; LE/EECS3311 3.00

EECS 4313 3.00 Software Engineering Testing

An introduction to systematic methods of testing and verification, covering a range of static and dynamic techniques and their use within the development process. The course emphasises the view that design should be carried out with verification in mind to achieve overall project goals. Topics to be covered include understanding the importance of systematic testing; understanding how verification is an integral part of the development process and not a bolt-on activity; understanding the strengths and weaknesses of particular techniques and be able to select appropriate ones for a given situation; black box and white box testing; Unit level testing techniques and practical exercises; mutation testing, domain testing, data flow and control flow testing; Coverage criteria; static analysis techniques (including program proof tools such as the Spark

Examiner or ESC/Java); Higher level testing (integration, system, performance, configuration testing etc.); testing tools and instrumentation issues; the testing of object oriented programs; testing non-functional properties of high integrity systems; worst case execution times, stack usage; hazard directed testing; software fault injection, simulation and hardware testing techniques; management issues in the testing process; planning, configuration management; quality assurance; controlling the test process; inspections reviews, walkthroughs and audits; influence of standards; regression testing.
All too often software is designed and then tested. The real aim must be to take a more holistic view, where design is carried out with verification in mind to achieve overall project goals. We shall take a fairly liberal view of testing. This includes various automated and manual static analysis techniques. In addition, we shall show how increased rigor at the specification stage can significantly help lower-level testing.

Learning outcomes for the course:

- Outline the objectives and limits of testing.
- Describe the strengths and weaknesses of the techniques discussed in the course.
- Select appropriate testing techniques for a given situation.
- Develop test harnesses for large software systems.
- Compile issue reports that are clear and complete.
- Produce quality written reports describing their testing.

Prerequisites: General prerequisites; LE/EECS3311 3.00

EECS 4314 3.00 Advanced Software Engineering

This course goes into more detail about some of the software engineering techniques and principles presented in earlier courses, and introduces advanced aspects of software engineering that are not addressed elsewhere. Topics to be covered include software process and its various models and standards (CMMI, ISO 9001); software architecture, i.e., the structure of data and program components that are required to build a software system. Examples include distributed and component-based architectures; Model Driven Engineering and the use of software description languages; software metrics, such as metrics for software quality, software design metrics, as well as testing and maintenance metrics; project management concepts on coordinating people and products; cost estimation and project scheduling for large software systems; risk management and mitigation; software configuration management (software evolution, change management, version and release management); emerging technologies, such as security engineering, service-oriented software engineering, and aspect-oriented software development.

Learning outcomes for the course:

- Derive models of software systems and express them in a language such as UML.
- Understand the differences between different types of software architecture
- Apply metrics that estimate the quality, maintainability, and test adequacy of a software system.
- Derive cost estimation tables delineating the tasks to be performed, and the cost, effort, and time involved for each task.
- Identify risks associated with a given software project, and develop plans to mitigate and manage these risks.
- Manage software projects by identifying the sequence of tasks that will enable the project to complete in time, assigning responsibility for each task, and adapting the schedule as various risks become reality.

Prerequisites: General prerequisites; LE/EECS 3311 3.00

EECS 4315 3.00 Mission-Critical Systems
Building on the material in *System Specification and Refinement* (EECS3342) which is an introduction to mathematical modelling and refinement of systems using deductive methods, this course provides students with a deeper understanding of both *deductive* and *algorithmic* methods and tools for ensuring the safety and correctness of mission critical systems (e.g., medical devices such as pacemakers, nuclear reactors and train control systems). In addition to deductive techniques, the course treats algorithmic methods such as model-checking tools, specification languages such as temporal logic, table based specification methods, real-time systems, and the nature of software certification.

Learning outcomes for the course:

- Explain the importance of safety-, mission-, business-, and security-critical systems.
- Demonstrate knowledge of the importance of good software engineering practices for critical systems.
- Use rigorous software engineering methods to develop dependable software applications that are accompanied by certification evidence for their safety and correctness.
- Demonstrate knowledge of the method and tools using deductive approaches (such as theorem proving).
- Demonstrate knowledge of methods and tools for algorithmic approaches (such as model checking, bounded satisfiability) etc.
- Demonstrate knowledge of the theory underlying deductive and algorithmic approaches.
- Use industrial strength tools associated with the methods on large systems.

Prerequisite: General prerequisites; LE/EECS3342 3.00

EECS 4351 3.00 Real-Time Systems Theory (not offered in 2018/2019) (integrated with EECS5441 3.00)

In real-time computing systems the correctness of the system depends not only on the logical result of the computation but also on the time at which the results are produced. For example, a computer controlling a robot on the factory floor of a flexible manufacturing system must stop or turn the robot aside in time to prevent a collision with some other object on the factory floor. Other examples of current real-time systems include communication systems, traffic systems, nuclear power plants and space shuttle and avionic systems.

Real-time programs in many safety-critical systems are more complex than sequential programs or concurrent programs that do not have real-time requirements. This course will deal with the modelling, simulation, specification, analysis, design and verification of such real-time programs. The objective of the course is to expose the student to current techniques for formally proving the correctness of real-time behaviour of systems.

Topics covered may include techniques for expressing syntax and semantics of real-time programming languages; modelling real-time systems with discrete event calculi (e.g., Petri net and state machine formalisms); specification of concurrency, deadlock, mutual exclusion, delays and timeouts; scheduling of tasks to meet hard time bounds; CASE tools for analysis and design. At the end of the course the student will be able to model and specify real-time systems, design and verify correctness of some real-time systems.

Prerequisites: General prerequisites; LE/EECS3221 3.00

EECS 4352 3.00 Real-Time Systems Practice (integrated with EECS5442 3.00)

The key aspect that differentiates real-time systems from general purpose computing systems is the need to meet specified deadlines. Failure to meet the specified deadlines can lead to intolerable system degradation, and can, in some applications, result in catastrophic loss of life or property. For example, the computations in an aircraft collision avoidance system must be completed before specified deadlines to prevent a mid-air collision. Real-time system technologies are applied in telecommunication, signal processing, command and control, digital control, etc. Examples of applications of real-time system technologies that impact our daily lives include engine, vehicle stability, airbag and break mechanisms in cars, flight control and air-traffic control, and medical devices.

The course focuses on the technologies related to the design and implementation of real-time systems. Topics may include typical real-time applications; process models of real-time systems; scheduling technologies in real-time systems; design and implementation of real-time systems software; real-time systems hardware; real-time operating systems; real-time programming languages; inspection and verification methods for real-time systems.

Learning outcomes for the course:

- Describe typical applications of real-time systems.
- Describe key components of real-time systems and applications.
- Analyse the performance and correctness of real-time systems and applications.
- Explain methods and techniques that are used in the design and implementation of real-time systems and applications.
- Design and implement a simple real-time application.

Prerequisites: General prerequisites; LE/EECS3221 3.00

EECS 4401 3.00 Artificial Intelligence (integrated with EECS5326 3.00)

This is a second course in artificial intelligence that covers selected topics in this area such as: reasoning about action and planning, uncertain and fuzzy reasoning, knowledge representation, automated reasoning, non-monotonic reasoning and answer set programming, ontologies and description logic, local search methods, Markov decision processes, autonomous agents and multi-agent systems, machine learning, reasoning about beliefs and goals, and expert systems.

Prerequisites: General prerequisites; LE/EECS3401 3.00

EECS 4402 3.00 Logic Programming (not offered in 2018/2019)
(integrated with EECS5311 3.00)

Logic programming has its roots in mathematical logic and it provides a view of computation that contrasts in interesting ways with conventional programming languages. Logic programming approach is rather to describe known facts and relationships about a problem, than to prescribe the sequence of steps taken by a computer to solve the problem.

One of the most important problems in logic programming is the challenge of designing languages suitable for describing the computations that these systems are designed to achieve. The most commonly recognised language is PROLOG.

When a computer is programmed in PROLOG, the actual way the computer carries out the computation is specified partly by the logical declarative semantics of PROLOG, partly by what new facts PROLOG can infer from the given ones, and only partly by explicit control information supplied by the programmer. Computer Science concepts in areas such as artificial intelligence, database theory, software engineering knowledge representation, etc., can all be described in logic programs.

At the end of this course a student will be familiar with fundamental logic programming concepts and will have some programming expertise in PROLOG.

Prerequisites: General prerequisites; LE/EECS3401 3.00; one of LE/EECS3101 3.00 or LE/EECS3342 3.00

EECS 4403 3.00 Soft Computing (not offered in 2018/2019)

This course introduces soft computing methods, which, unlike hard computing, are tolerant of imprecision, uncertainty and partial truth. This tolerance is exploited to achieve tractability, robustness and low solution cost. The principal constituents of soft computing are fuzzy sets and logic, neural network theory, rough sets, evolutionary computing and probabilistic reasoning. The course studies the methods and explores how they are employed in associated techniques as applied to intelligent systems design. The basics of each technique will be discussed and applications will illustrate the strengths of each approach. The course is self-contained. Knowledge of mathematics, in particular basic probability and statistics, and familiarity with a high-level programming language is assumed. The class will have several programming/homework assignments, a presentation, a final exam and a final project.

Learning outcomes for the course:

- Knowledge of the terminology and concepts of soft computing;
- Insight into the possibilities and fundamental limitations of soft computing;
- Insight into the relative advantages and disadvantages of the major approaches to soft computing (fuzzy sets, rough sets. Evolutionary computing, neural networks, probabilistic reasoning and so on);
- Understanding of the basic methods and techniques used in soft computing;
- Skills in applying the basic methods and techniques to concrete problems in soft computing.

Prerequisites: General prerequisites; LE/EECS2011 3.00; LE/EECS2031 3.00

EECS 4404 3.00 Introduction to Machine Learning and Pattern Recognition (integrated with EECS5327 3.00)

Machine learning is the study of algorithms that learn how to perform a task from prior experience. Machine learning algorithms find widespread application in diverse problem areas, including machine perception, natural language processing, search engines, medical diagnosis, bioinformatics,

brain-machine interfaces, financial analysis, gaming and robot navigation. This course will thus provide students with marketable skills and also with a foundation for further, more in-depth study of machine learning topics.
This course introduces the student to machine learning concepts and techniques applied to pattern recognition problems in a diversity of application areas. The course takes a probabilistic perspective, but also incorporates a number of non-probabilistic techniques.

Learning outcomes for the course:

- To develop powerful pattern recognition algorithms using probabilistic modelling and statistical analysis of data.
- Identify machine learning models and algorithms appropriate for solving specific problems.
- Explain the essential ideas behind core machine learning models and algorithms.
- Identify the main limitations and failure modes of core machine learning models and algorithms.
- Program moderately complex machine learning algorithms.
- Manage data and evaluate and compare algorithms in a supervised learning setting.
- Access and correctly employ a variety of machine learning toolboxes currently available.
- Identify a diversity of pattern recognition applications in which machine learning techniques are currently in use.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; one of SC/MATH2030 3.00 or SC/MATH1131 3.00

EECS 4411 3.00 Database Management Systems

This course is the second course in database management. It introduces concepts, approaches, and techniques required for the design and implementation of database management systems. Topics covered may include query processing; transactions; concurrency control; recovery; database system architectures; distributed databases; object-oriented databases.

Learning outcomes for the course:

- Describe and apply indexing techniques used in relational database systems.
- Use relational operators to design query evaluation plans.
- Assess query evaluation plans for the purpose of query optimization.
- Perform database administration for a given workload.
- Describe and classify modern non-relational database systems.

Prerequisites: General prerequisites; LE/EECS2011 3.00; LE/EECS2021 4.00; LE/EECS2031 3.00; LE/EECS3421 3.00

EECS 4412 3.00 Data Mining

Data mining is computationally intelligent extraction of interesting, useful and previously unknown knowledge from large databases. It is a highly inter-disciplinary area representing the confluence of machine learning, statistics, database systems and high-performance computing. This course introduces the fundamental concepts of data mining. It provides an in-depth study on various data mining algorithms, models and applications. In particular, the course covers data pre-processing, association rule mining, sequential pattern mining, decision tree learning, decision rule learning, neural networks, clustering and their applications. The students are required to do programming assignments to gain hands-on experience with data mining.

Learning outcomes for the course:

- Explain the terminology and concepts of data mining.
- Evaluate the possibilities and fundamental limitations of data mining.
- Evaluate the relative advantages and disadvantages of major approaches to data mining.
- Demonstrate the basic methods and techniques used in data mining.
- Apply the basic methods and techniques to actual problems in data mining

Prerequisites: General prerequisites; LE/EECS3101 3.00; LE/EECS3421 3.00; one of SC/MATH2030 3.00 or SC/MATH1131 3.00

EECS 4413 3.00 Building E-Commerce Systems

A study of the technical infrastructure that underlies electronic commerce on the internet. The foundational concepts are presented through a series of projects that use an industrial-strength framework on the server side, standard-compliant technologies on the client side, and a variety of messaging protocols on the network side. Best practices, security concerns, and performance issues are emphasized throughout. A good knowledge of Java is assumed. Familiarity with the upper networking protocols, HTML, JavaScript, and SQL is helpful but not required.

Learning outcomes for the course:

- Develop an appreciation of the pieces that make up the web landscape and how these pieces interact with each other.
- Build a complete web application that incorporates session management, database access, and analytics on the server side, and page formatting and interactivity on the client side.
- Build restful web services that interact with Ajax-powered client apps using a variety of transport protocols for data transfer.
- Become familiar with, and adhere to, best practices and design patterns to ensure code maintainability, interoperability, and scalability, and to minimize exploitable vulnerabilities.
- Learn how to build complex applications collaboratively through building abstractions and APIs, naming conventions, documentation, and organizing.
- Compare and contrast existing frameworks and approaches and develop an insight into the tectonic forces that are driving the trends.

Prerequisites: General prerequisites; LE/EECS2011 3.00

EECS 4414 3.00 Information Networks (not offered in 2018/2019)

Information networks are effective representations of pairwise relationships between objects. Examples include technological networks (e.g., World Wide Web), online social networks (e.g., Facebook), biological networks (e.g., Protein-to-Protein interactions), and more. The study of information networks is an emerging discipline of immense importance that combines graph theory, probability and statistics, data mining and analysis, and computational social science. This course provides students with both theoretical knowledge and practical experience of the field by covering models and algorithms of information networks and their basic properties. In addition, analysis of information networks provides the means to explore large, complex data coming from vastly diverse sources and to inform computational problems and better decisions. Topics include: basic graph theory, network measurements, network models, community detection, graph partitioning, link analysis, link prediction, information cascades & epidemics, influence maximization, network ties, recommendation systems, mining graphs, and connections to problems in the social sciences and economics.

Learning outcomes for the course:

- Understand basic graph theory concepts and network analysis.
- Develop a quantitative and qualitative intuition of the role of information networks in social and technological systems.
- Develop a quantitative and qualitative intuition of network analysis methods and algorithms.
- Understand the tools and concepts aimed at analysing the structure and dynamics of networks arising from the interplay of human behavior, socio-technical infrastructures, information diffusion and biological agents.
- Obtain practical experience working with software and tools for large-scale social and information network analysis.

Prerequisites: General prerequisites; LE/EECS3421 3.00; LE/EECS3101 3.00; one of SC/MATH 2030 3.00 or SC/MATH 2930 3.00

EECS 4415 3.00 Big Data Systems
Storing, managing, and processing datasets are foundational to both computer science and data science. The enormous size of today's data sets and the specific requirements of modern applications, necessitated the growth of a new generation of data management systems, where the emphasis is put on distributed and fault-tolerant processing. New programming paradigms have evolved, an abundance of information platforms offering data management and analysis solutions appeared and a number of novel methods and tools have been developed. This course introduces the fundamentals of big data storage, retrieval, and processing systems. As these fundamentals are introduced, exemplary technologies are used to illustrate how big data systems can leverage very large data sets that become available through multiple sources and are characterized by diverse levels of volume (terabytes; billion records), velocity (batch; real-time; streaming) and variety (structured; semi-structured; unstructured). Topics include: software frameworks for distributed storage and processing of very large data sets, MapReduce programming model,

querying of structured data sets, column stores, key-value stores, document stores, graph databases, distributed stream processing frameworks.

Learning outcomes for the course:

- Assess the limitations of traditional architectures for storing, retrieving and processing data.
- Design alternative big data architectures and assess their tradeoffs.
- Work with systems and tools for storing, retrieving and processing big data.
- Understand novel methods and algorithms for processing big data (as they appear in technical reports and other literature) and port/transfer them in technology solutions and applications.
- Develop novel methods and software programs that store, retrieve and process big data in multiple scenarios.

Prerequisites: General prerequisites; LE/EECS3421 3.00; LE/EECS3101 3.00

EECS 4421 3.00 Introduction to Robotics (integrated with EECS5324 3.00)

The course introduces the basic concepts of robotic manipulators and autonomous systems. After a review of some fundamental mathematics the course examines the mechanics and dynamics of robot arms, mobile robots, their sensors and algorithms for controlling them. A Robotics Laboratory is available equipped with a manipulator and a moving platform with sonar, several workstations and an extensive collection of software.

Learning outcomes for the course:

- Explain the basic terms and key concepts behind mobile robots and robot arms.
- Analyse the forward and inverse kinematics of mobile robots and robot arms.
- Apply basic mathematical techniques to solve problems in robotics.
- Analyse the effects of noise in robotic navigation.
- Use techniques to reduce the effects of noise in robotic navigation.
- Develop software to solve problems in robotics.

Prerequisites: General prerequisites; SC/MATH1025 3.00, SC/MATH1310 3.00, LE/EECS2031 3.00

EECS 4422 3.00 Computer Vision (integrated with EECS 5323 3.00)

This course introduces the fundamental concepts of vision with emphasis on computer science. In particular the course covers the image formation process, colour analysis, image processing, enhancement and restoration, feature extraction and matching, 3-D parameter estimation and applications. A Vision Laboratory is available equipped with cameras, workstations, image processing software and various robots where students can gain practical experience.

The course includes 12 hours of supervised lab sessions.

Learning outcomes for the course:

- Explain the basic terms and key concepts in computer vision.
- Apply basic mathematical techniques to solve problems in computer vision.
- Develop software to solve problems in computer vision.
- Analyse the effects of noise in computer vision algorithms.
- Use techniques to reduce the effects of noise in computer vision algorithms.

Prerequisites: General prerequisites; SC/MATH1025 3.00; SC/MATH1310 3.00; LE/EECS2031 3.00

EECS 4425 3.00 Introductory Computational Bioinformatics

This course is intended to provide an introduction to theoretical and practical foundations necessary to a computer scientist working in the bioinformatics field. Topics covered in the course include molecular biology for computer scientists; sequencing analysis algorithms; NCBI, National Centre for Biotechnology Information, BioJava, Java tools for processing biological data; Biological databases; Phylogenetic trees; microarray data analysis for gene expression.

Prerequisites: General prerequisites; LE/EECS2011 3.00

EECS 4431 3.00 Advanced Topics in 3D Computer Graphics
(integrated with EECS5331 3.00)

This course discusses advanced 3D computer graphics algorithms. Topics may include direct programming of graphics hardware via pixel and vertex shaders, real-time rendering, global illumination algorithms, advanced texture mapping and anti-aliasing, data visualization, etc. Real-time image generation (rendering) techniques and direct programming of graphics hardware via pixel and vertex shaders are technology that is increasingly used in computer games. Furthermore, these are also often used for computationally intensive applications as graphics hardware has far surpassed the raw computational power of traditional CPU's. Advanced texture mapping and anti-aliasing algorithms are used to create better quality images, that show less digital artefacts. Global illumination algorithms are used to generate images that are indistinguishable from real photos. Such images are used in the film industry, architecture, games, and lighting design.

Visualization is a key technology for dealing with large data volumes, which are typically generated by computational simulations (weather forecasting, aerodynamic design, etc.) or by sensor networks (satellites, geology, etc.). In these fields, visualization in graphical form enables humans to understand the vast amounts of data and the phenomena that they represent.

Two-hour lab sessions will be held during 6 weeks of the course.

Learning outcomes for the course:

- Describe and use advanced hardware platforms, such as shader-based streaming processors, in computer graphics systems.
- Describe approaches that solve the rendering equation, and model advance concepts related to global illumination.
- Explain and measure performance issues related to interactive computer graphics applications.
- Implement multi-pass rendering approaches to solve rendering and image processing problems.
- Model scenes in real-time that include reflective and refracted surfaces.
- Compose CG elements into images of captured real-world environments using image based lighting.
- Dynamically generate new geometric elements within the graphics pipeline using geometry and tessellation shaders.

Prerequisites: General prerequisites; LE/EECS2021 4.00; LE/EECS3431 3.00

Course Credit Exclusions: COSC4331 3.00

EECS 4441 3.00 Human Computer Interaction (integrated with EECS5351 3.00)
This course introduces the concepts and technology necessary to design, manage and implement interactive software. Students work in small groups and learn how to design user interfaces, how to realize them and how to evaluate the end result. Both design and evaluation are emphasized. The topics of this course will be applied in practical assignments and/or group projects. The projects will consists of a design part, an implementation part and user tests to evaluate the prototypes.

Prerequisites: General prerequisites; LE/EECS3461 3.00
Course Credit Exclusions: COSC4341 3.00

EECS 4443 3.00 Mobile User Interfaces

Students learn how to design, implement, and test user interfaces for contemporary mobile devices such as smart phones and tablet computers. Design issues will consider the limits and capabilities of human sensory, perceptual, cognitive, and motor behaviour and how these impact human interaction with mobile technology. Many common features in mobile devices are not available in desktop computer systems and, consequently, are not taught in other courses. As well as a graphical display, the devices of interest for this course include touch input (including multi-touch and finger pressure sensing), device position and motion sensing via accelerometers and gyroscopes, environmental sensors, actuators for vibrotactile output, audio capture, and camera input. The development of user interfaces for these devices is complex since the target system and development systems are, by necessity, different. Thus, the course will include instruction on the development environment including the use of a debugger, simulator, and emulator, and connecting the target and development systems via a physical or wireless link, and uploading and downloading files, including the installation of application software.
The course format is weekly 3-hour lectures and 2 hours of lab exercises every other week.

Prerequisites: General prerequisites; LE/EECS3461 3.00

EECS 4452 3.00 Digital Signal Processing: Theory and Applications

Digital signal processing (DSP) has become the foundation of various digital systems and communication and entertainment applications in today's computer era. This course consists of two parts. The first part introduces students to the fundamental DSP concepts, principles and algorithms. The second part covers some important DSP-based applications in the real world.

This course is designed to cover most of DSP theory and algorithms and some selected important DSP applications. In lab projects, students will design and implement some DSP systems in selected application areas, such as speech and audio processing or image processing, by using either particular DSP hardware (such as TMS 320 series DSP chips) or software simulation, to get hands-on experience of DSP system design.

Learning outcomes for the course:

- Explain the operations and key components in signal processing systems.
- Calculate time and frequency representations of digital signals.
- Analyse the characteristics of linear digital systems.
- Design and implement various types of digital signal processing systems according to specifications.
- List and describe selected real-world applications of signal processing techniques.

Prerequisites: General prerequisites; LE/EECS3451 3.00 or LE/EECS3602 4.00

EECS 4461 3.00 Hypermedia and Multimedia Technology (not offered in 2018/2019)

The course focuses this year on the design and implementation of hypermedia presentation systems. "Hypermedia" refers to the non-linear organization of digital information, in which items (such as a word in a text field or a region of an image) are actively linked to other items. Users interactively select and traverse links in a hypermedia presentation system in order to locate specific information or entertainment, or to browse in large archives of text, sound, images, and video. Well-structured hypermedia gives users a way of coping with the "navigation" problem created by availability of low-cost, fast access, high-density storage media.

Students will be expected to familiarise themselves quickly with the Macintosh interface and basic features of the operating system. Students will be asked to schedule themselves for at least six hours/week lab time in the Department's Multimedia Lab, as the course work will involve a significant amount of exploration and development of multimedia/hypermedia materials. Students will be divided into small teams with specific responsibilities for individual exploration and programming tasks assigned in connection with the course topics. Tasks may take the form of constructing presentations, prototype applications, or the programming of useful scripts. The teams will be asked to write short reports on their work that will be presented in class.

Prerequisites: General prerequisites; LE/EECS3461 3.00

EECS 4462 3.00 Digital Audio

This course introduces the basic principles of digital audio, and presents several of its applications. Students will learn the physics of sound and the human auditory system, how analog audio is converted to digital, and the properties of different digital audio formats. They will also get hands-on experience with creating audio plugins for digital audio workstations, as well as creating sound engines for games.

Learning outcomes for the course:

- Explain the nature of sound from a physics perspective
- Describe the human audio perception system
- Explain how digital audio is created and stored, as well as the pros and cons of different audio formats
- Describe the difference between Midi-based and sample-based audio
- Create audio processing plugins for modern digital audio workstations
- Create a sound engine for a simple game

Prerequisites: General prerequisites; LE/EECS2031 3.00

EECS 4471 3.00 Introduction to Virtual Reality

This course introduces the basic principles of Virtual Reality and its applications. The necessary hardware and software components of interactive 3D systems as well as human factors are discussed. The material is reinforced by practical assignments and projects.

The scheduled lab sessions involve practical experimentation with virtual environments and will support the development, presentation and demonstration of a comprehensive student design project. Two-hour lab sessions will be held alternate weeks in the Virtual Reality lab.

Learning outcomes for the course:

- Explain the perceptual capabilities and limitations of the human user and how they relate to an effective virtual environment. Identify important human factors concerns.
- Describe major application areas and technologies for virtual reality systems.
- Implement interactive computer systems, including input, simulation and display, using the virtual reality metaphor.
- Simulate physical environments in a compelling manner using immersive technology, event driven simulation, animation and physical modelling.
- Work in small teams to develop, test and demonstrate a comprehensive student design project.
- Design, document and present a complex project in an effective manner covering both technical and application domain issues.

Prerequisites: General prerequisites; SC/MATH1025 3.00; SC/MATH1310 3.00; LE/EECS2021 4.00; LE/EECS2031 3.00; LE/EECS3431 3.00

EECS 4480 3.00 Computer Security Project

This is a capstone project course for computer security students. The students engage in a significant research and/or development project that has major computer security considerations. This is a required course for Computer Security students.

Students who have a project they wish to do need to convince the course director that it is appropriate for course credit. They also need to find a faculty member that agrees to supervise the project. Alternatively, students may approach a faculty member (typically, one who is teaching or doing research in computer security) and ask for project suggestions. For students that are not able to find a suitable project through the above means, the course director is responsible for preparing appropriate projects. Any of the projects may be individual or team projects at the discretion of the course director (coordinator).

Whatever the origin of the project, a “contract” is required. It must state the scope of the project, the schedule of work, the resources required, and the criteria for evaluation. The contract must be signed by the student, his/her project supervisor, and the course director. A critical course component that must be included in the contract is a project presentation to take place after the project is finished. The course director will arrange the presentation sessions, and students and their faculty supervisors are required to participate. The presentations will have a typical length of 15-20 minutes, and will be evaluated by the individual supervisor, the course director and at least one more faculty member.

Learning outcomes for the course:

- Apply the knowledge they have gained in other computer security courses to a real-world system.
- Understand the computer security challenges faced by the information technology industry.
- Articulate the questions that a particular area of research in computer security attempts to address.
- Prepare a professional presentation that outlines the contributions they made to the project and the knowledge they acquired.

Prerequisites: Restricted to students in the Computer Security degree. Students must have passed 40 LE/EECS credits. Permission of the 4080/4480 coordinator is required.

Course Credit Exclusions: LE/EECS 4080 3.00, LE/EECS 4081 6.00, LE/EECS 4082 6.00, LE/EECS 4084 6.00, LE/EECS4088 6.00, LE/EECS 4700 6.00

EECS 4481 4.00 Computer Security Laboratory

This course provides a thorough understanding of the technical aspects of computer security. It covers network, operating system, and application software security. Computer laboratory projects provide exposure to various tools in a hands-on setting. Topics to be covered include Access Control - Identification, authentication, and authorization; trust management; Network Security - attacks, intrusion detection, auditing and forensics, firewalls, malicious software, packet monitoring and other tools/techniques for finding network security related problems.; Operating System Security - threats, vulnerability, and control, password management, accounts and privileges; Application Software Security - design of secure systems, evaluation, Java security, buffer overflows, database security, client-side and server-side securities, tamper resistant software and hardware, finding vulnerabilities, developing patches, patch distribution; Thinking Evil (understand the enemy so that you can design better software and systems)—how to build a virus, Trojan, worm, (how to detect them and break them); real-world vulnerability detection.

Learning outcomes for the course:

- Explain popular attacks targeting operating systems and networks.
- Write intrusion detection signatures.
- Design and deploy security controls (e.g., firewalls, honeypot, password policy).
- Discover vulnerabilities in computer systems.
- Analyse and alter network traffic.
- Explain and identify different types of buffer overflow.
- Design secure software applications.

Prerequisites: General prerequisites; LE/EECS3221 3.00; LE/EECS3214 3.00

EECS 4482 3.00 Computer Security Management: Assessment and Forensics

This course examines the organizational policy and management aspects of computer security. It covers topics such as policies, procedures, and standards related to access and use, compliance and privacy, risk management and incident response. Specific topics covered include Information Security Fundamentals - basic terminology and concepts: confidentiality, integrity, availability, authentication, auditing, information privacy, legal aspects, etc; Security Policies - security plan (how to develop one), policies, procedures, and standards, acceptable use policies, compliance and enforcement, policy-based management systems (how they work, examples); Access Controls - physical, technical, and data access, biometrics; Risk Management - risk analysis and threat quantification, contingency planning, disaster recovery; Incident Response - response methods, emergency response teams, forensics principles and methodology, computer crime detection and investigation; Inappropriate Insider Activity: the problem, the cure?; Ethics.

Learning outcomes for the course:

- Describe the importance of the manager's role in securing an organization's use of information technology and explain who is responsible for protecting an organization's information assets.
- Explain the unified contingency plan approach.
- Discuss the process of developing, implementing, and maintaining, various types of information security policies.
- List and describe the functional components of an information security program List and describe the typical job titles and functions performed in the information security program.
- Discuss the components of a security education, training, and awareness program and explain how organizations create and manage these programs.

Prerequisites: Any 12 credits at the 3000-level

EECS 4491 3.00 Simulation and Animation for Computer Games (not offered in 2018/2019)

This course presents the conceptual foundation of simulation and animation methods used in the Digital Media industry, including computer games. Students will get an understanding of the theory and techniques behind making objects "move" in an interactive environment. The course covers all aspects, including manual animation, (semi-) automatic animation through simulation of the movement of linkages and body-parts, animation through recordings of real motions (motion capture), the simulation of physics for rigid bodies, liquids, gases, plants, and deformations, as well as combinations of these methods. Topics to be covered include principles of "Classic" Animation; spaces, Transformations, and Rotations; interpolation Methods; interpolation-Based Animation; kinematic Linkages; Inverse Kinematics; motion capture; Physically Based Animation; modelling liquids & gases; modelling and animating human figures; facial animation; modelling behaviour; special models for animation.

Learning outcomes for the course:

- various interpolation methods to move objects in a virtual environment in a believable manner.
- a system to simulate rigid, animated objects
- movement of animated figures consisting of multiple limbs

- examples of physically based animations, such as particles,
- liquids and deformations.
- simple methods for motion capture and interpolation for captured motion data.

Prerequisites: General prerequisites; LE/EECS3431 3.00; SC/MATH1310 3.00

EECS 4611 4.00 Advanced Analog Integrated Circuit Design
The course presents advanced design techniques for the realization of high-performance analog integrated circuits in modern technology. In particular, high-speed and low-noise amplifiers are targeted along with certain nonlinear components employed in discrete-time signal processing and narrowband applications. The features and limitations of modern semiconductor devices are presented and the means of abstracting these to compact models applicable to hand design are outlined. A number of feedback system analysis and design techniques are presented (compensation, dominant-pole, root locus) and applied to wide-band amplifier realization. The origin of noise in electronic components is reviewed and means of mitigating it through circuit design explained. Key nonlinear analog device behaviour are highlighted and applied to the design of critical nonlinear circuitry such as switches, comparators, mixers, output stages, and power amplifiers. Means by which robust designs treat component variability are addressed.

Learning outcomes for the course:
- Characterize and abstract the behaviour of modern electronic devices into compact models for the purpose of analog circuit design.
- Understand and design stabilization strategies for wide-band amplifiers.
- Understand and implement techniques for the realization low-noise amplifiers.
- Analyse and design high-speed analog switching amplifiers.
- Verify the behaviour of high-performance designs using simulation and experimental techniques.

Prerequisite: General prerequisites; LE/EECS3611 4.00

EECS 4612 4.00 Digital Very Large Scale Integration

The objective of this course is to introduce the students to the design of large-scale integrated semiconductor digital systems and to promote the application of their acquired knowledge in the synthesis of their own integrated system. The course lectures review, introduce, and detail the key components of these systems (transistors, gates, arithmetic units, hardware description languages, memories, I/O elements) while the initial laboratory sessions give the students hands-on exposure to the construction of a large digital system. This laboratory experience spans the essential constituents of the VLSI design hierarchy including standard cell layout, datapath arrangement, control unit design, floorplanning, place and route, and verification. Students also gain substantial experience with industry-standard VLSI CAD tools as part of their laboratory work. The remaining lab sessions are used to help the students apply the knowledge and techniques acquired in the course to design and complete a custom VLSI system of their own choosing. Besides presenting and detailing purely engineering science concepts related to physical circuit understanding and analysis the course presents classical design concepts critical to successful system realization including VLSI design methodology, circuit design trade-offs, manufacturing considerations and the economics of IC fabrication.

Learning outcomes for the course:

- Demonstrate knowledge of the contemporary VLSI design methodology and be able to cogently describe the main steps of a VLSI design flow.
- Demonstrate the ability to approximate the physical size of an integrated VLSI circuit given sufficient structural and technological information.
- Discuss contemporary VLSI design IC technologies (fabrication, structure, and applications) and be able to mathematically articulate how they work and what their electronic capabilities and drawbacks are.
- Demonstrate the ability to simplify advanced IC technologies into basic circuit models capable of providing sufficient approximations for first-order digital circuit analysis.

- Demonstrate the ability to optimize a variety of digital circuits for energy-delay performance.
- Demonstrate the ability to design and synthesize semi-custom hardware sub-systems such as state machines, memories, logic arrays, busses, etc. in a modern IC technology from a library of standard cells.
- Demonstrate the ability to apply hardware design languages for the realization of complete integrated VLSI systems through hands-on practice with non-trivial (greater than 10,000 gates) examples and custom designs.
- Refining the ability to analyse the performance of VLSI systems using established criteria such as latency, throughput, operational energy consumption as well as newly emphasized issues such as power management and clock distribution.
- Demonstrate an understanding of the means of verifying the operation of complex digital ICs.
- Gaining expertise in state-of-the-art custom IC design tools.
- Create a practical VLSI project work plan and meet designated deadlines towards that plan.
- Effectively communicate a personal design in person and in the form of a written report.

Prerequisites: General prerequisites; LE/EECS2200 3.00; LE/EECS2210 3.00; LE/EECS3201 4.00

EECS 4613 4.00 Power Electronics
This course focuses on the basic operating principles of the power conversion using advanced electronic devices. The structure and characteristics of switching devices are first reviewed. Basic semiconductor devices used in power electronics circuits are discussed. Fundamental power converters such as AC/DC rectifiers, DC/DC switching converters and voltage source DC/AC inverters are studied. Resonant power converters and inverters are introduced. Weekly laboratory/tutorial. Topics to be covered include switching devices in power electronics, single-phase AC/DC uncontrolled rectifiers, three-phase uncontrolled AC.DC rectifiers; single-phase AC/DC controlled rectifiers; DC/DC converters; isolated DC/DC converters; single-phase DC/AC converters;

resonant DC/DC converters and inverters; applications of power electronics.

Learning outcomes for the course:

- Understand the basic operating principles of power conversion in power electronics
- Understand the operating principles of AC/DC rectifiers, DC/DC converters, DC/AC inverters
- Demonstrate the ability to mathematically compute average and RMS value, average power, power factor and total harmonics distortion
- Demonstrate the ability to mathematically compute switching losses, conduction losses, and power efficiency
- Demonstrate the ability to determine the suitable control method for the corresponding converter or inverter.
- Derive mathematical equations that characterize the voltage gain of basic DC/DC converters.
- Demonstrate the ability to draw steady-state operating waveforms of AC/DC rectifiers, DC/DC converters, DC/AC inverters
- Understand the features and drawbacks of different isolated DC/DC converters
- Understand the basic operating principles of resonant power conversion.
- Demonstrate the ability to design simple power electronic circuits for real-life applications.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; LE/EECS2210 3.00
Co-requisites: LE/EECS3603 4.00

EECS 4614 4.00 Electro-Optics (not offered in 2018/2019)

The objective of this course is to provide the students with an understanding of the basic concepts of light-matter interaction and control of light propagation with emphasis on modern technological applications. It continues from the notions of the electromagnetic nature of light and Maxwell’s equations, and applies these concepts to classical applications in optics such as light propagation at boundaries with metals and dielectrics. Light-matter interactions are discussed throughout the course and fundamental concepts of semiconductor physics are introduced as needed. The course contents span a broad range of topics to provide knowledge of the most important building blocks of electro-optic systems used, for example, in imaging, display, telecommunication and sensing applications.

Engineering design will also be emphasized throughout the course in the description of practical applications of the technologies presented and through practical design problems in the lab. The laboratory modules (three-hour weekly) include practical and/or simulation & design activities in topics such as: polarization of light, waveguides, optical fibers, lasers and optical modulators.

Learning outcomes for the course:

- Demonstrates understanding of light propagation in matter and interfaces, and can numerically simulate the behaviour in multilayers.
- Demonstrates understanding of light polarization and guiding, and can design means to control them.
- Comprehends light absorption and emission, and the necessary conditions for light amplification.
- Comprehends the basics of semiconductor lasers operation and design.
- Understand how to combine the constraints for higher harmonic generation through practical design.
- Demonstrates the ability to combine several electro-optical building blocks by designing waveguide-based or waveguide-coupled lasers, photoreceivers and modulators.
- Demonstrates practical hands-on skills in applying in the lab the principles thought in class.

Prerequisites: General prerequisites; LE/EECS 2030 3.00 or LE/EECS 1030 3.00; LE/EECS 3604 or SC/PHYS4020; LE/EECS2210 or SC/PHYS3150

EECS 4622 4.00 Introduction to Energy Systems

This is an introductory course for energy systems. It covers the basic construction, modelling and analysis techniques in electricity generation, transmission and distribution. Topics to be covered include steady-state modelling and performance of transmission lines; single line power diagrams and per-unit systems; admittance and impedance bus matrix for power networks; load flow analysis; fault (short circuit) analysis; dynamic stability; distributed power systems.

Learning outcomes for the course:

- Have a clear definition about the basic concepts such as complex power, single line diagrams, per unit quantities.
- Power system model to include different components –generator – transformer –static and –dynamic loads.
- Differentiate the steady-state modelling for different types of transmission lines (short, medium, and long) and (overhead versus underground cables).
- Analyse the steady-state performance of transmission lines.
- Formulate the admittance and impedance bus model for power networks.
- To perform power-flow studies by different methods and use it to perform system design and operations using digital computer programs.
- Perform analysis for symmetrical and unsymmetrical faults.
- Understand the different forms for power system dynamic such as steady state-transient and dynamic stability.
- Understand the different topologies of distribution systems and introduce the concept of active distribution systems with high penetration of distributed and renewable energy resources.

Prerequisites: General prerequisites; LE/EECS2200 3.00; LE/EECS3603 3.00; SC/PHYS2020 3.00

EECS 4641 4.00 Introduction to Medical Devices and Biological Instruments

This course builds on the foundation in measurement techniques by developing the students' understanding of electrophysiological sensing systems and biosensors used within the medical and biological fields. This course applies the knowledge of electronic circuits and systems techniques to the development of medical devices and biological instruments. Background in electronic circuit design, basic knowledge of human physiology and body system, and basic knowledge of cellular and molecular biology are required before taking the course. In the design of each medical device or biological instruments, students are also introduced the related theoretical and practical issues with a focus on needs assessment, creativity, and innovation as they seek to identify market opportunities. A detailed list of topics covered within this course is as follows.

Three lecture hours and three laboratory hours each week. The mandatory laboratory applies the design and implementation concepts to representative medical devices and biological instruments chosen from described devices.

Learning outcomes for the course:

- Develop an understanding of electrical measurement techniques including EEG, EMG and ECG.
- Develop an understanding of biosensors and point-of-care techniques used for early detection or control of diseases such as diabetes and cancer.
- Develop an understanding of biological instruments such as cell counter, biological electrical activities' recording and stimulation
- Be able to analyse and design biomedical measurement systems.
- Develop an understanding of the safety and regulatory issues pertinent to medical devices.
- Develop an understanding of advanced technologies such as microfluidics and Lab-on-Chips.

Prerequisites: General prerequisites; LE/EECS 2210 4.0; LE/EECS 3215 4.0; BIOL 1000 3.0*

* **Note**, a new science elective course covering the basic topics of biological and health sciences is being designed by Kinesiology in consultation with EECS to be offered as one of the prerequisites of this course**.** In the short term BIOL1000 3.00 will serve as an adequate substitute prerequisite until the specialized course is mounted.

EECS 4642 4.00 Medical Imaging Systems (not offered in 2018/2019)
This course provides an introduction to several of the major imaging modalities including X-ray, ultrasound and magnetic resonance technologies. The students learn the fundamental, operation and basic design of medical imaging instrumentations. This course applies the

classical knowledge of physics, signal and systems techniques to the development of various medical imaging technologies. Background in the design of medical devices; electronic circuit, and systems; basic knowledge of human physiology, and basic knowledge of physics are required before taking the course. Topics include the physics of radiography; fundamental of X-ray projection radiography and X-ray computed tomography (CT); fundamental of ultrasound imaging techniques and ultrasound imaging systems; and introduction of nuclear medical resonance and MRI system.

Three lecture hours and three laboratory hours each week. The mandatory laboratory applies the analysis of medical images captured from available medical imaging instrumentations in York University.

Learning outcomes for the course:

- Explain the physics of medical imaging systems.
- Design circuits and systems for medical imaging systems.
- Explain medical image processing techniques and clinical applications of medical imaging systems
- Employ medical imaging systems and apply low complexity image processing techniques.
- Compare and evaluate advanced medical imaging technologies.

Prerequisites: General prerequisites; LE/EECS2210 3.00; LE/EECS3602 4.00 or LE/EECS3451 4.00

EECS 4643 4.00 Biomedical Signal Analysis

The sensing, detection, processing, estimation and classification of biological signals are core to biomedical engineering and important in human-machine interaction, biological engineering, agriculture and other applications. This course applies discrete-time and continuous-time theory to the processing of biomedical signals that can be used to infer the physiological state of a living organism. Background in linear systems theory in both discrete and continuous time is required before taking this course. Topics to be covered include origins of biomedical signals, deterministic signal processing, stochastic processing, time domain analysis and feature extraction; classical spectral analysis; noise reduction and filtering; time-frequency and wavelet analysis; multidimensional and

multivariate signal processing; electrical activity of the body; biomedical signals; signal classification; applications to medical devices.
Three lecture hours and three laboratory hours each week. The mandatory laboratory applies concepts to representative biomedical signal processing problems.

Learning outcomes for the course:

- Understand the processes relating biological signals to their physiological origins, their stochastic nature and the need for signal processing.
- Extend existing signal processing skills to multivariate, multichannel and stochastic signals.
- Assess, choose and implement methods to extract features of interest in biological signals or to classify normal and abnormal signals.
- Implement appropriate signal processing algorithms to filter and enhance noisy biological signals.
- Design signal processing solutions for important application domains such as physiological monitoring, diagnosis, and human-machine interfaces (e.g. for prosthetics).

Prerequisites: General prerequisites; LE/EECS 3602 4.00 or LE/EECS 4452 3.00

EECS 4700 6.00 Digital Media Project
This is an honours thesis course in Digital Media. Although a course coordinator will be assigned to the course, the bulk of the course will take place through the interaction between a supervisor and the group of students. After two organizational meetings in September, the students will work with their supervisor directly. The course requires an initial project proposal that will be submitted to and approved by the supervisor and the course coordinator (director). This is, in essence, a contract for the project to follow. The supervisor will evaluate the performance of the students in early January. The format of this evaluation will vary from project to project, but the requirements of this evaluation will be specified in the original project proposal. At the beginning of the course, the course director (coordinator) will establish a date and format for the public presentation of all Digital Media projects. Normally held between reading week and the third last week of term, this presentation will normally consist

of either a short public oral or poster presentation of the project. (The actual format may change from year to year.) All of the faculty associated with the Digital Media program will be invited to attend this presentation. The individual supervisor will mark this presentation and the final report due at the end of the term.

The actual nature of the project will vary from student to student. Projects will involve the design, implementation and evaluation of a Digital Media work. The expectation is that all projects will involve creation of a digital media artefact and possibly also the evaluation of human interaction with the product, including an analysis of these results in the presentation and final report. For projects that will involve significant subject testing and performance evaluation, it is expected that a complete draft implementation of the system will be available by January. Supervisors may be faculty from either the Department of Electrical Engineering and Computer Science or the Faculty of Fine Arts or the Communication Studies program of the Division of Social Science, Faculty of LA&PS.

Marking Scheme:
Mid-term evaluation: 30%
Public presentation evaluation: 30%
Final report: 40%

Prerequisites: Only open to students in the final year of the Digital Media program.
Course Credit Exclusions: LE/EECS4080 3.00, LE/EECS4081 6.00, LE/EECS4082 6.00, LE/EECS4084 6.00, LE/EECS4088 6.00, LE/EECS4480 3.00