The official most up-to-date version is found at URL
http://www.cse.yorku.ca/undergrad/csCalendars.html

Preface 3
The Department 3
Faculty 4
CSAC and CEAB Accreditation 5
A Note on Terminology 5
Programs Offered by the Department 5
The Computer Science Program 6
Streams in Honours Computer Science Programs 7
The Computer Security Program 8
The Digital Media Program 8
International Programs 9
iBSc and iBA 9
The International Dual Degree BSc Specialised Honours Program 9
The Computer Engineering Program 11
The Software Engineering Program 11
The Electrical Engineering Program 12
Degree Requirements 13
Courses on Offer in 2014-15 13
Admission to Programs 13
Computer Science and Computer Security Programs 13
Digital Media Program 14
Electrical, Computer and Software Engineering Programs 14
Graduate Programs in Computer Science and in Engineering 14
Technology Internship Program 15
Out of Major Elective Courses - Computer Science and Computer Security Programs 15
The Service Program 16
Recent Academic Changes 17
Student Clubs 17
The Student Ombuds Service 18
Computer Facilities 18
Computer Use Policy 20
Awards 21
Academic Policies 22
Appeal Procedures 24
Grading System 25
Courses Offered by the Department 26
Course Descriptions: 1000-Level 27
Course Descriptions: 2000-Level 42
Course Descriptions: 3000-Level 49
Course Descriptions: 4000-Level 65
Access to Courses 96
Normal Order of Study 97

Prerequisites for Computer Science Courses.....99
Degree Program Checklists.....105

Preface

In choosing to study Computer Science, Computer Security, Digital Media, Electrical Engineering, Computer Engineering or Software Engineering you have chosen a career in exciting and rapidly changing disciplines. As a professional in one of these fields, you may become involved in many of the great changes in the future, for the computing and engineering disciplines will play a central role in these changes.

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. For that reason we would strongly encourage you to select, in addition to the required courses prescribed in the programs offered by the department, courses outside the disciplines of Computer Science or Engineering in areas where you will broaden your knowledge of societal issues. One way to do this is to select isolated courses that catch your interest; however, a more productive approach is to consider taking a concentration of courses in a different area or possibly designing a second major or minor in addition to your primary major—although double majoring or major/minoring is not open in the Computer Security, Digital Media or the Engineering Programs.

In general, in planning your course selection, you should be thinking ahead and asking yourself not only which technical/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.

The Department

Electrical Engineering and Computer Science Department
1003 Lassonde Building (LAS)
York University
4700 Keele Street
Toronto, Ontario M3J 1P3
http://www.cse.yorku.ca/

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

Vice Chair (Science):	Tel. (416) 736-5334
George Tourlakis	Email: ug@cse.yorku.ca
Vice Chair (Engineering):	Tel: (416) 736-5334
Robert Allison	Email: enquiries@cse.yorku.ca
Vice Chair (Graduate):	Tel. (416) 736-5053
U. T. Nguyen	Email: enquiries@cse.yorku.ca
Chair:	Tel. (416) 736-5053
Richard Wildes	Fax: (416) 736-5872

Faculty

	Telephone Extension	email @cse.yorku.ca
Aboelaze, Mokhtar	40607	aboelaze
Allison, Robert	20192	allison
Amanatides, John *	44782	amana
An, Aijun	44298	aan
Asif, Amir	70128	asif
Baljko, Melanie	33348	mb
Cercone, Nick	55053	nick
Cribb, Peter	70127	peterc
Datta, Suprakash	77875	datta
Dymond, Patrick	33948	dymond
Eckford, Andrew	70152	aeckford
Edmonds, Jeff	33295	jeff
Elder, James	66475	jelder
Faloutsos, Petros	40630	pfal
Farag, Hany	33844	hefarag
Ghafar-Zadeh, Ebrahim	44646	egz
Godfrey, Parke *	66671	godfrey
Gotshalks, Gunnar	33350	gunnar
Gryz, Jarek	70150	jarek
Hofbauer, John *	70125	hofbauer
Hornsey, Richard	33265	hornsey
Jenkin, Michael *	33162	jenkin
Jiang, Hui *	33346	hj
Jiang, Jack	33939	zmjiang
Kant, Mariana	70117	mkant
Lam, John		johnlam
Lesperance, Yves *	70146	lesperan
Lian, Yong (Peter)	44647	peterlian
Ma, Burton	77885	burton
Mackenzie, Scott	40631	mack
Magierowski, Sebastian	44652	magiero
Mirzaian, Andy	70133	andy
Nguyen Uyen	33274	utn
Ostroff, Jonathan	77882	jonathan
Pisana, Simone		pisana
Roumani, Hamzeh	66146	roumani
Roventa, Eugene (*Emeritus*)		roventa
Ruppert, Eric	33979	ruppert
Spetsakis, Minas	77886	minas
Stachniak, Zbigniew	77877	zbigniew
Stuerzlinger, Wolfgang	33947	wolfgang
Toptsis, Anestis	66675	anestis
Tourlakis, George	66674	gt
Tsotsos, John	70135	tsotsos
Tzerpos, Vassilios	33341	bil
van Breugel, Franck	77880	franck
Vlajic, Natalija	77878	vlajic
Wallis, Anthony (*Emeritus*)		wallis
Wharton, Michael	33978	michael
Wildes, Richard	40203	wildes
Xu, Jia	77879	jxu

* On Sabbatical

CSAC and CEAB Accreditation

The Computer Science Accreditation Council (CSAC) has accredited all Computer Science honours (major) programs offered by the Department that have already graduated students. The Computer Engineering specialised honours BEng program has already graduated students and is accredited by the Canadian Engineering Accreditation Board (CEAB).

The Computer Science Accreditation Council is an autonomous body established by the Canadian Information Processing Society (CIPS), while the Canadian Engineering Accreditation board is established by Professional Engineers Canada. The purpose of accreditation is to identify those institutions that offer computer programs worthy of recognition. The objectives of the accrediting bodies are:

- To formulate and maintain high educational standards for Canadian universities offering computer and information science programs, and to assist those institutions in planning and carrying out education programs.
- To promote and advance all phases of computer 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 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 the Canadian Engineering Accreditation Board (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/ and on the Engineers Canada website at http://www.engineerscanada.ca/

A Note on Terminology

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

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. Electrical Engineering
4. Computer Engineering
5. Software Engineering
6. Digital Media

For detailed information you are advised to first read the appropriate sections of the York University Undergraduate Calendar (click on the related York University's web page http://calendars.registrar.yorku.ca). Secondly, read this supplemental Calendar, and thirdly, see an advisor in the Department.

The Computer Science Program

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 most subjects in each of Lassonde School of Engineering, LA&PS and Faculty of Science, leading 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 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 for other programs thereby achieving greater depth of study. However, a breadth in education is maintained by the requirement of a significant number (30 credits) of non-EECS and non-MATH courses.

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 or BSc program, normally completed in three years of study, requires 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. Similarly, all three Engineering programs offered by EECS have a common first year so that all programs get a common strong foundation in mathematics, computing and engineering principles, but also so that students have the option to make an informed choice of the program they will follow from the end of year one to graduation. All computer science programs are structured in such a way that a student who embarks on a 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 (with the exception of the minor) are accredited by the CSAC.

The degree requirements for the various Computer Science degree programs and Streams, as well as for Computer Security, Digital Media, Electrical, Computer, and Software Engineering offered by the Department are listed at the end of this calendar (as a URL link in the case of the on-line version of this document).

Streams in Honours Computer Science Programs

The Specialised Honours and Honours programs may be taken with a specified focus (specialization) or *Stream*. The streams provide a mechanism for recognising on your transcript a particular emphasis or focus. Available streams currently include:

1. Communication Networks
2. Intelligent Systems
3. Interactive Systems
4. Software Development

Each stream *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 Honours program), as well as a full year (6 credit) 4000–level project, or honours thesis as it would be called in some universities.

[1] *In December 2005 the Senate of York University has approved, with effective date of implementation April 3, 2006, an amendment that allows students to "**proceed on warning**" if they fail to meet the gpa of 5.0. The **minimum** cumulative gpa **required** are 4.00 between 0-23 credits; 4.25 between 24-53 credits; 4.80 between 54-83 credits; 5.00 beyond 83 credits.*

The Computer Security Program

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, database, 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.

The Digital Media Program

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) as well as 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 2-year common core of courses followed by two distinct specialisation Streams, one named *Digital Media Development* while the other *Digital Media Arts*. Either stream involves an approximately equal number of courses from the Department of Electrical Engineering and Computer Science and the Faculty of Fine Arts. 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).

For more information see the URL:

http://futurestudents.yorku.ca/program/digital-design

or the program requirements here:

http://www.cse.yorku.ca/undergrad/csCalendars.html (2014-15)

International Programs

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.
For more information see the URL:

http://www.cse.yorku.ca/csprosp_students/undergrad/iBSc/index.html

Since 2003 the Department has maintained a successful International Summer School program, mounting courses in partnership with departments in Germany, Greece and Poland.

For more information see the URL:

http://www.cse.yorku.ca/cscurrent_students/undergrad_students/international/index.html

The International Dual Degree BSc Specialised Honours Program

This is our newest international program, which started in the fall 2011:

- An international program of study at York University, the Hochschule Bonn-Rhein-Sieg (BRSU) and the University of Crete (UoC) that equips the graduate with professional credentials in North America and in Europe.
- Two degrees 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.

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). This limited-space program will be of interest to students with high academic standing as measured by a cumulative GPA of 6.00 or higher over all computer science (EECS) courses

completed ("major gpa") *by the time that students have completed approximately a total of 60 credits 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, subject to the aforementioned GPA requirement, to continue their studies as York international exchange students in the European Union (EU) *for a full year of study*. This exchange placement will be divided between BRSU in the Fall term and UoC in the Winter term. At UoC, the students will complete a mandatory *research internship* component and an *undergraduate thesis*, and may, optionally, also take specialised computer science courses.

The thesis and internship activities will both be conducted in English.

Study at two universities in distinct geographic and linguistic/cultural settings adds value to the exchange and broadens the learning experience. The program of study is precisely regulated as dictated by the need 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 4th year of study at York University to fulfil their BSc Specialised Honours degree requirements and thus also graduate with the York University Honours degree. York International administers all exchanges under this program in collaboration with the International Offices in BRSU and UoC.

Reciprocally, BRSU students will spend a full year of study at York to conclude their 3rd year BRSU requirements 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* in lieu of the general education requirements of the Lassonde School of Engineering, 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* in lieu of the current general education requirements of the Lassonde School of Engineering, as follows: They must complete at York University 6 further non-science credits in addition to 12 credits in language and culture courses.

The Computer Engineering Program

This is a Specialised Honours BEng (Bachelor of Engineering) 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.

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.

For more information see also the URL: http://lassonde.yorku.ca/computer-engineering/

The Software Engineering Program

This Specialised Honours BEng (Bachelor of Engineering) Degree Program commenced in September 2011.

- It is a professional degree - your entry to the engineering profession.
- It offers core knowledge in software engineering that closely matches IEEE-ACM software engineering curriculum guidelines.
- It offers specializations in mobile communications, databases, human-computer interfaces, security, and networks and net-centric computing.
- It develops teamwork, communication skills and encourages an industrial internship.

Software Engineering 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.

Software engineers need professional skills to develop complex mission critical systems with design architectures that support reliability, extensibility and reusability. Thus, software engineers must have an understanding of systematic design processes of large-scale integrated systems including project planning, requirements analysis, design, coding, testing, configuration management, quality assurance, and documentation.

Building on existing strengths in Computer Science and Computer Engineering, York's new 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 with an industrial internship extending anywhere from four to sixteen months.

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. See also http://lassonde.yorku.ca/software-engineering-beng

The Electrical Engineering Program

This is our newest Specialised Honours BEng (Bachelor of Engineering) Degree Program that commenced in September 2013. 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.
See also http://lassonde.yorku.ca/electrical-engineering/

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

Course requirements fall into two or three broad categories:

1. Those required for the major, i.e. computer science and mathematics courses; and for the Digital Media degree, fine arts cultural studies and social science courses.
2. Those required for the second major 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 Department also provides degree checklists that itemize the course requirements in a succinct and clear way (hopefully!). 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 included at the end of the hardcopy version of this supplemental calendar and also at this link

http://www.cse.yorku.ca/cscurrent_students/undergrad_students/index.html

under the heading "Program Information".

Courses on Offer in 2014-15

The course schedule for Summer 2014 and FW 2014-15 is found on the department Undergraduate page for continuing students

http://www.cse.yorku.ca/cscurrent_students/undergrad_students/index.html

under the heading "Course Information".

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 to include, computer science will need to meet the following *minimum* requirement:

- 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 programs (minimum OCGPA of 4.00 (grade of C) is needed to transfer into the Bachelor degree programs).[2]
- *Must* meet *either* the mathematics Admissions requirement, *or* the prerequisite alternatives (2) or (3) in the prerequisite of EECS1020. Qualifications for entry cannot be mixed (entry is decided either on Admission qualifications alone, or on *one* of the two alternatives above).

Once transferred to a computer science 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 Computer Science and in Engineering

Admission to the graduate program is highly competitive. The ideal preparation for graduate studies in Computer Science or Computer Engineering is the completion of the Specialised Honours Program in Computer Science, or in Computer Engineering, or in an equivalent degree (*that includes senior level courses in theoretical computer science*). Your grade point average in the last two years should be at least B+ to enter the competition for admission. Of course, the higher your grades the more likely you will be a successful candidate. For more information, please visit
http://www.cse.yorku.ca/cscurrent_students/grad_students/index.html

[2] *All courses listed in your York University transcript will be included in the calculation of your cumulative grade point average.*

Technology Internship Program

The Technology Internship Program (TIP) offers qualified undergraduate Computer Science, Computer Security, Computer Engineering, Software Engineering, Electrical Engineering and Digital Media students the opportunity to take part in a program that alternates academic studies with related work experience in either the private or public sectors. There is considerable flexibility in the duration of individual Internships. The length of an Internship can vary from four to sixteen months. During the Internship placement students earn a salary typical of entry-level positions in the IT sector.

Students in the BEng, BA Honours, BSc Honours, iBSc, iBA are eligible to apply in year three. Students enrolled in the Internship option are required to enrol in EECS3900 0.00 (Internship Co-op Term) in *each term* of their internship—ENG3900 0.00 if the are in a BEng degree program. For administrative reasons we have a separate course, EECS3980 0.00 associated with Internships of Computer Security majors.

The Department has formed a partnership with the Career Centre of York University to offer better services to students who are interested in the Internship Program. The Career Centre assists students seeking internship employment and also assists employers wishing to hire York University Internship students. Internship students receive assistance in identifying relevant and interesting internship opportunities, formulating the employer application package and sharpening their interview skills. Students are placed at a wide range of companies including IBM, RIM, Sun Microsystems, Platform, Workbrain, Ontario Lottery and Gaming Commission, CIBC, Toronto Hydro, Ontario Power Generation, and Global Matrix.

For additional information please visit the link http://www.yorku.ca/careers/intern/
or e-mail intern@career.yorku.ca

See also the EECS3900 0.00 and EECS3980 0.00 and ENG3900 0.00 descriptions in this supplemental calendar.

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 to

name just 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 (one of the sciences, for example, whose hierarchical course structure ensures that none of the courses therein can be taken in isolation), 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 outside the discipline elective courses, all honours programs require at least 30 credits from non-EECS and non-MATH courses.

The Service Program

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 Computer Use: Web and Database Systems
EECS1560 3.00 Introduction to Computing for Math and Statistics
EECS1570 3.00 Introduction to Computing for Psychology

EECS1520 3.00 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 3.00 is an introduction to computer programming and may be taken as preparation for EECS1020 3.00 or for EECS2501 1.00, if the student lacks background in this area. EECS1550 3.00 offers a practical way of learning the basics of how information is specified, acquired, and managed using database technology. EECS1541 3.00, EECS1560 3.00 and EECS1570 3.00 are directed towards Physics and Astronomy, MATH/Stats and Psychology majors, respectively.

Students taking the 1500 series courses are not enabled to take the 2000-level EECS courses for majors without prior successful completion of EECS1020 3.00 and EECS1030 3.00.

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.

Recent Academic Changes

All Undergraduate CSE courses now have—as of Fall 2014—a new "*rubric*" (course prefix): EECS.

1. Program Changes (2014/15)

   - Digital Media (BA Specialised Honours program) is now—for all students admitted for Fall 2014—*streamed*, that is, one must choose either the Digital Media Development, or the Digital Media Arts Streams. The first two years of study are common to the two streams. For a detailed list of course requirements see the Undergraduate page of EECS here http://www.cse.yorku.ca/cscurrent_students/undergrad_students/index.html or here http://lassonde.yorku.ca/electrical-engineering-computer-science/undergrads-courses-0 under the heading Program Information.
   - Computer, Software and Electrical Engineering Programs now have a *revised* (from the one valid through 2013/14) common first year effective Fall 2014. For a detailed description of the new common first year, and a small number of changes to the second year of three Engineering Programs, see the section on Degree Program Checklists.

4. New Courses and Course Changes FW2014/15

   - New EECS1011 3.00 Computational Thinking through Mechatronics.
   - New EECS1021 3.00 Object Oriented Programming from Sensors to Actuators.
   - New EECS/MATH1028 3.00 Discrete Mathematics for Engineers.
   - New EECS2602 4.0 Signals and Systems in Continuous Time.
   - ENG2200 3.00 and ENG2210 3.00 have changed rubric. They now are EECS2200 3.00 and EECS2210 3.00 respectively.
   - CSE2550 1.00 and CSE2560 1.00 have been discontinued.
   - CSE3341 3.00 has been discontinued. In all programs where CSE3341 3.00 is required, EECS3342 3.00 is a direct substitute.

Student Clubs

The York University Computer Club (YUCC) is an organization of students who share an interest in computing. They nominate students to serve on Department committees, sponsor informational and social events and facilitate communications among students and faculty members. They can be reached by electronic mail at yucc@yucc.yorku.ca

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

The York University chapter of Engineers Without Borders—http://www.yorku.ewb.ca—helps people in developing communities gain access to the technology they need to improve their lives. In the past, EWB@York repaired Pentium-based computers and shipped them to Iraq for female NGOs. Summer Internships include three international and one local placement.

The York University Rover Team (YURT). From their Profile page, http://yorku.collegiatelink.net/organization/roverteam/about
"YURT is a group of enthusiastic undergraduate and graduate students from a wide array of disciplines who advanced rover prototypes to compete in NASA's Lunabotics Mining Competition and Mars Society's University Rover Challenge…YURT is a two-time winner at the University Rover Challenge and placed first at the 2012 Canadian Innovation Nation Robotics Competition." YURT's credo is that "solving technological hurdles of today will lead to a better tomorrow." Contact: info@yuroverteam.com or drop by at Room 002 in Petrie Building (basement level) to join.

The Women in Computer Science and Engineering (WiCSE) supports and promotes women in Computer Science and in Engineering. The objectives of WiCSE include: (i) providing a support network for female computer science and engineering students; (ii) implementing a mentoring program to assist them in the preparation of applications for scholarships, bursaries and summer jobs, providing guidance in career development and post graduate education; and (iii) improving the "climate" for women and help student attraction and retention. They can be contacted through their website http://www.cse.yorku.ca/csWiCSE/index.html.

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 holds 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.

Computer Facilities

Undergraduate students who are registered to 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 store or print their course related files, use electronic mail facilities, create their own web sites. Students access the Unix or Windows workstations in the laboratories through scheduled sessions or first come first serve basis. The accounts can also be accessed remotely by dial-up, through the Internet via secure connection, or from other designated laboratories on campus. Select laboratories are equipped with printing facilities. First year engineering 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 Robotics and Vision, the Digital Systems, the Wave and Power Engineering, the Software Engineering, the Networking and Computer Security, the Integrated Signal Processing and Multi-Media, and the Virtual Reality Laboratories.

- The Robotics and Vision Laboratory consists of two CRS robot arms, an autonomous mobile robot, fifteen Unix workstations equipped with multimedia hardware including monocular and stereo video cameras and audio facilities.
- The Digital Systems Laboratory provides 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 Windows workstations, embedded microcontroller boards, logic analysers, oscilloscopes and other electronic test equipment to provide students with hands-on experience on design and implementation of digital and embedded systems
- The Software Engineering Laboratory consists of a project meeting area and a work area with Unix 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 Virtual Reality Laboratory was established to support the study of modern virtual reality systems. It consists of a variety of specialised hardware displays and tracking devices including a large screen passive stereoscopic display, two Phantom Omni haptic devices, immersive audio displays, two head mounted displays and a number of magnetic and inertial motion tracking devices.
- The Integrated Signal Processing and Multi-Media Laboratory consist of a number of Digital Signal Processing boards, each contains a processor, memory, and I/O channels with A/D and D/A capabilities. The laboratory is equipped with function generators, oscilloscopes and power supplies. Windows workstations are also available for program development, and simulation.
- The Wave and Power Engineering Laboratory is equipped with tools to provide an environment where the engineering students experiment and learn the fundamentals of Electrical Engineering.

- The Networking and Computer Security Laboratory consists of Windows workstations equipped with specialised software tools and hardware equipment for networking and computer security courses.
- The Attack Laboratory 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. Students can have administrator privileges and work on complex network topologies, something not possible in a physical laboratory.
- The Digital Media Laboratory consists of workstations equipped with video capturing devices and software suites that are tailored to the development of interactive, media-rich applications. This includes various compilers and development environments (e.g., Java, Python, Cycling 74's Max/MSP, Eclipse) as well as video- and image-, and audio-manipulation suites. The laboratory is used for classroom instruction, tutorials, student work, and student evaluation (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 computers around the world via the Internet. All laboratories have access points to provide wireless Internet access resources.

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 of 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 Department computer system 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 Award

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 best students in the Department.

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 fo 7.00 (B+).

Academic Policies

Advising

Academic advising is available on an individual basis in the Department. 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, assistance with getting the most favourable degree audit possible consistent with their academic profile.

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.[3] Written information and program checklists are provided to assist you in making

[3] *The BSc and BA degrees follow university-wide standards.*

appropriate choices. It is recommended that *you take advantage of advising opportunities to answer any questions you may have*.

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

Academic 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, integrating it with their work, and perhaps more significantly, reducing some of the drudgery in producing a piece of 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 policies
http://www.yorku.ca/secretariat/policies/document.php?document=69
for further information and the penalties when academic dishonesty occurs.

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 director. 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.

Moving to New Program Requirements and New Prerequisites

Computer Science and Engineering disciplines constantly respond and adapt to technological and theoretical progress. To ensure that our students graduate with current degree programs that 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—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 terms. For example: starting in fall 2001 you follow the 2001-02 program requirements; starting in winter 2002 or summer 2002 you also follow the 2001-2002 program requirements.
- If program requirements change but you did not have either interruptions in your studies—or program changes—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://www.cse.yorku.ca/undergrad/csCalendars.html
- 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* to courses or to groups of 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 normally *are effective starting with the term immediately following their approval.*

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 see
http://www.registrar.yorku.ca/grades/reappraisal/

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.

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 parenthesis 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.

Courses Offered by the Department

Prerequisites

Almost all 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.

Prerequisites are enforced in every term. Independently of their major, students who are enrolled in EECS courses for which they do not meet the prerequisite will be de-enrolled and notified by email. This prerequisite checking 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.

Prerequisites may include both specific courses and, at the 3000- and 4000-level, also the requirement of a cumulative GPA of 4.5 or higher computed over all EECS *major* courses. "General Prerequisites" is a term used to describe prerequisites that apply to (almost) every course at a particular level. We use it simply to avoid having to repeatedly specify the same thing! Thus, for example, for most 3000-level courses, "EECS2011 3.00 and the EECS GPA of 4.5 or higher" are *the general prerequisites*.

Associated Course Fees

All courses have an associated fee of $10.00, with the following exceptions: The courses associated with the Technology Internship Program, namely, EECS3900 0.00, EECS3980 0.00 and ENG3900 0.00 have a fee of $475.00.

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

Course Weights

Courses normally meet for three class hours a week for one term (these are 3 credit–courses whose numbers end in "3.00"). Some courses have required supervised labs every week (e.g., EECS1020 3.00 and EECS1030 3.00). Catalogue numbers are assigned to the labs rather than the lectures and students use the *Registration and Enrolment Module* (REM) to enrol by selecting an appropriate lab. Other courses have a similar registration system and lab requirements, but the associated labs are of sufficient duration per week to entail a 4.00-credit weight for the course (e.g., EECS2021, 2041, 3201, 3215, 3451 and 4481 are "4.00-credit courses"). Some of the 3.00 courses at the 2000 and 3000 levels have optional tutorials. All EECS courses put heavy demands on the student's time by requiring the completion of take-home assignments or projects.

Service Courses

The nomenclature applies to courses that the department mounts for the degree program needs of other majors, e.g., Mathematics and Statistics, Psychology, Biology, Physics, Chemistry, etc. Such courses are identified by a second digit 5 (e.g. 1520, 1530, 1540, 1541, 1550, 1560, 1570, 2501). These can be taken as electives in the major programs offered by the department but *not* as major EECS credits. The grades from such courses are not included in calculating the major grade point average that the general prerequisites speak of (see above under the "prerequisites" heading).

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 (if at all). 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 will consist of 12 one-hour lectures spread over two terms. The first lecture will be organizational in nature. The remaining 11 lectures will be comprised by 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 also 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.

Main Topics

1. The Computing Environment: Workspace, built-in commands, the debugger, unit testing, plots, etc.
2. Variables and Expressions: Types, operators, precedence, roundoff errors
3. Control Structures: Selection and Iteration
4. Encapsulation: Script files and functions
5. Computational Thinking: Process-based problem solving, unit tests as specification

Soft Computing Skills

1. Reasoning about algorithms
2. Tracing program
3. Test-driven Development

Applications

1. General Science and Mathematics
2. Engineering applications derived from the various engineering programs in the Faculty.

Learning Objectives for the Course:

By the end of the course, the students will be able to:

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

Prerequisites: None

Course Credit Exclusions: EECS1541 3.00

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. The detailed list of topics includes

1. Proof techniques (without using a formal system)
   - proof by contradiction
   - proof by cases
   - proving implications
   - proving statements with quantifiers
   - mathematical induction on natural numbers
2. Naïve set theory
   - proving that one set is a subset of another
   - proving equality of two sets
   - basic operations on sets (union, intersection, Cartesian product, power sets, etc.)
   - cardinality of sets (finite and infinite)

strings

3. Functions and relations
   review of basic definitions (relation, function, domain, range, functions, 1-1 correspondence, function composition, closures of relations, etc.)
   equivalence relations
4. Asymptotic notation
   big-O, big- Ω, big- Θ notation
   proving f is in O(g), proving f is not in O(g)
   classifying functions into a hierarchy of important classes, e.g.,
   $$O(1),\ O(\log n),\ O(\sqrt{n}),\ O(n),\ O(n^2),\ O(n^{O(1)}),\ O(2^n)$$
5. Recursive definitions and solving recurrences
   recursive definitions of mathematical objects
   solving simple recurrences
   bounding divide-and-conquer recurrences of the form $f(n) = af(n/b) + g(n)$, for constants a and b.
   using structural induction on recursively defined objects
6. Sums
   summation notation
   computing and bounding simple sums
7. Elementary graph theory
   basic definitions of graphs
   proving simple facts about graphs
   trees

Prerequisites: MATH1190 3.00, or two 4U courses including MHF4U (Advanced Functions)
Course Credit Exclusions: MATH2320 3.00

EECS 1020 3.00 Introduction to Computer Science I

Many processes can be viewed as a sequence of interactions between a client who requests a service and an implementer who provides it. The concerns of these two parties, albeit complementary, are completely separate because one deals with the "what" while the other deals with the "how". It is widely recognized that separating these concerns leads to reliable, scalable, and maintainable software. Based on this, EECS1020 deals exclusively with the client who needs to be able to look for services; read their API (Application Programming Interface) specifications; create programs that use them; and determine if they are operating correctly relative to their specifications. Topics include delegation and contracts, encapsulation and APIs, aggregation and the collections framework, and inheritance and polymorphism. The course emphasizes the software development process and introduces elements of UML (Unified Modelling Language) and software engineering. Three-hour lectures and weekly laboratory sessions.

The course uses the Java programming language throughout. Its assessment is based on a series of programming exercises and a number of written tests. The two

components have approximately equal weights and are intended to measure the student's understanding of theoretical concepts and ability to build applications.

This course is an introduction to the discipline; 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 computer science. The course is intended for prospective computer science and computer engineering majors, i.e. those with a well-developed interest in computing as an academic field of study and with strong mathematical, analytical and language abilities; it is not intended for those who seek a quick exposure to applications or programming (for this purpose any of EECS1520, EECS1530 or EECS1540 would be more appropriate).

Warning: The work for this course includes a substantial number of exercises that require problem analysis, program preparation, testing, analysis of results, and documentation and submission of written reports. The course is demanding in terms of time, and requires the student to put in many hours of work per week outside of lectures.

Recommendation: You will benefit if you have prior practical experience with programming as well as using a computer. Students who wish to take a one-course exposure to the practical aspects of computing should consider enrolling in EECS1520 3.00 and EECS1530 3.00 instead (see the following descriptions).

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

(1) (New high school curriculum): Two 4U Math courses including MHF4U (Advanced Functions), with no grade below 65%.

(2) Completion of 6.00 credits from York University MATH courses (not including AK/MATH1710 6.00 or courses with second digit 5) with a grade point average of 5.00 (C+) or better over these credits;

(3) Completion of AK/MATH1710 6.00, or 6.00 credits from York University mathematics courses whose second digit is 5, with an average grade not below 7.00 (B+).

Strongly Recommended: Previous programming experience; for example, a high school programming course or EECS1530 3.00.

Course Credit Exclusions: ITEC1620 3.00, EECS 1021 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 two hours of weekly lectures are complemented by three-hour long weekly labs.

The lab hardware is chosen so that it can interface with a variety of languages (including MATLAB and Java); has several analog-to-digital and digital-to-analog converters; can control external power supply, and has a form factor suitable for undergraduate labs.

Learning Objectives for the Course:

By the end of the course, the students will be able to:

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

Prerequisites: EECS1011 3.00

Course Credit Exclusions: EECS1020 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. Three-hour long weekly lectures and two hours of mandatory tutorials per week.

The detailed list of topics includes

1. Propositional logic, truth tables. Applications:
    - Building various switching circuits using OR, AND, NAND, etc., gates (cf. for example, from Rosen 9.3).
2. Sets (union, intersection, Cartesian product, power sets, etc.), cardinality of sets (finite and infinite), strings.
3. Functions and relations (domain, range, 1-1, into, onto, 1-1 correspondence, function composition, closures of relations, etc.) equivalence relations.

4. Predicate logic, properties of quantifiers (e.g., understanding the connection between $\forall$ and $\exists$ via $\neg$), proving statements with quantifiers.
5. Proof techniques including proof by contradiction, proof by cases, proving implications (assuming the antecedent and proving the conclusion).
6. Mathematical induction on natural numbers, and structural induction on recursively defined objects, such as formulae, trees. Applications:
    - Proving that simple (loop and recursive) programs behave as intended (this entails several case studies not just one example).
7. Basic counting, subsets of a set, binomial notation, binomial theorem; sum and product notation. Applications:
    - Estimating security of passwords (i.e., computing the number of words coming from a given alphabet, and interpreting this in terms of likelihood of guessing a password correctly).
8. Elementary graph theory, including trees and spanning trees. Applications:
    - Circuit analysis (finding the fundamental [independent] cycles of a circuit by finding the *spanning tree* of the underlying graph);
    - Storing and retrieving data efficiently (sort trees).
    - Huffman coding (cf. for example, Rosen p.701);
9. Finite state machines with and without output, as tables and as state diagrams. Applications:
    - Arguing (by induction) that a given machine behaves correctly according to a given specification.
    - Using automata to assess correctness of simple concurrent processes.
    - Using automata for text lexical analysis.

Learning Objectives:

By the end of the course, the students will be expected to be able to:

- Prove any propositional formula that is a tautology, using truth tables or syntactic proof techniques such as resolution.
- Show why a propositional formula is not a theorem (not a tautology).
- Build simple switching circuits using OR, NAND, AND, etc., gates.
- Manipulate expressions involving intersection, union and set-theoretic difference, and deduce whether different such expressions represent the same set.
- Express relations abstractly as a subset of a Cartesian product of sets, and construct a partition from an equivalence relation (and vice versa).

- Specify the domain and codomain of a given function, determine whether a function is 1-1 or onto (or both), and apply these concepts to infer whether composition of functions and inverse functions are well defined.
- Compute the cardinality of finite sets of objects constructed according to the various set-theoretic operations of union, intersection, Cartesian product and power set.
- Derive formulas of the cardinality of sets of objects depending on a parameter, such as the number of strings of length n from a given alphabet, or the number of ordered or unordered sequences of length n.
- Exploit the pigeonhole principle in cardinality arguments.
- Prove or disprove as the case may be simple formulas in quantified logic.
- Translate English mathematical statements into predicate logic formulas.
- Be able to correctly form the negations of “for some x A(x) is true” and “for all x A(x) is true”
- 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; prove properties of well-formed formulas (e.g., equal number of left and right brackets).
- Be able to reason about graphs and (binary) trees and use them in several examples
    - To demonstrate an understanding of locating the fundamental cycles of an electrical circuit
    - Use a tree to efficiently store and retrieve information.
    - Be able to show simple properties of trees (examples: relation between number of nodes and edges; relation between number of nodes and height)
    - Use trees to find the Huffman codes of symbols of an alphabet that have probabilities of occurrence attached to them.
- Construct simple finite automata based on a specification and argue (typically by induction) that they behave exactly as specified.

- 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” in a compiler, which finds numbers and identifier names in a computer program as well as keywords such as “if”, “then”, “else”, “begin”, “end”, etc.)
- Construct simple automata to assess the correctness of simple concurrent programs.

The grade weight distribution of the course components is as follows:

15% - Assignments (biweekly)
15% - Quizzes (following the assignments)
30% - Midterm (in-class)
40% - Final Exam (scheduled by the Registrar office)

Recommended Text: Discrete Mathematics and Its Applications, by Kenneth H. Rosen, ISBN: 0073383090; Publisher: McGraw-Hill.

Prerequisites: MHF4U and MCV4U

Course Credit Exclusions: EECS/MATH1019 3.00, MATH2320 3.00

EECS 1030 3.00 Introduction to Computer Science II

This course continues the separation of concern theme introduced in EECS1020. While EECS1020 focuses 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/non-utilities, 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. Three lecture hours and weekly laboratory sessions.

Learning Objectives:

By the end of the course, the students will be expected to be able to:

- 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) analyze the running time of (recursive) algorithms.

Lab tests and in-class tests are integral parts of the assessment process in this course.

Prerequisites: EECS1020 3.00 or EECS1720 3.00
Course Credit Exclusions: ITEC2620 3.00

EECS 1520 3.00 Computer Use: Fundamentals

This course is appropriate for students who are not majoring in Computer Science or Computer Engineering, 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; database; 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 EECS1020 *3*.00/1030 3.00: *Introduction to Computer Science.*

Prerequisites: None
NCR Note: No credit will be retained if this course is taken after the successful completion of or simultaneously with EECS1020 3.00.
Note: This course counts as elective credits towards satisfying Faculty degree requirements but does not count as Computer Science major credits.

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* Computer Science or Computer Engineering majors. However, those who wish to major in Computer Science but lack programming background may use it as preparation. Students who plan to major in Computer Science must also take EECS1020 3.00 and EECS1030 3.00. This course does not count as a Computer Science major credit.

Prerequisites: None
Course Credit Exclusions: EECS1540 3.00
NCR Note: No credit will be retained if this course is taken after the successful completion of or simultaneously with EECS1020 3.00 or 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 and students in the BA Applied Math program.

Note: This course is not open to any student who has passed or is taking EECS1020 3.00. This course counts as elective credits towards satisfying Faculty degree requirements but does not count as Computer Science major credits.

Suggested reading:

- Nyhoff and Leestma, *Fortran 77 for Engineers and Scientists*, 3rd Edition, Maxwell Macmillan.
- Keiko Pitter et. al., *Every Student's Guide to the Internet* (Windows version), McGraw-Hill, 1995.

Prerequisites: None.
Course Credit Exclusions: EECS1530 3.00
NCR Note: No credit will be retained if this course is taken after the successful completion of or simultaneously with EECS1020 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.

Topic Overview:

An integrated procedural programming, data analysis, and data visualization platform such as MATLAB (or its open-source equivalent OCTAVE) will be used to introduce computational elements. Topics will include:

- Overview of the platform and computational accessories
- Fundamentals of the platform, including operational syntax
- Data types (including cell arrays)
- Data file input/output
- Data statistics
- Basic and advanced plotting of data, including surface and contour plots

- Procedures and control structures, including syntax, conditional evaluation statements (IF-ELSE), and loop programming (nested FOR loops, WHILE loop)
- Code vectorization
- User-defined functions
- Advanced functions (including nested and recursive functions and sorting)
- Simple matrix methods (systems of linear equations)
- Random numbers, and simple Monte-Carlo simulation (area estimation with statistical error assessment)
- Data acquisition

Physics faculty members will assist in choosing applications from first-year physics to use, such as recursive stepping through motion in two dimensions using a non-Calculus finite-difference approach.

Two hours of lectures and three hours of Lab exercises weekly. To be offered once per year in the Winter Semester.

Prerequisites: MATH1013 3.00 or equivalent.
Co-requisites: PHYS 1010 6.00, PHYS 1410 6.00, or PHYS 1420 6.00; and MATH 1021 3.00 or MATH 1025 3.00.
Course Credit Exclusions: EECS 1560 3.00, EECS1570 3.00

EECS 1550 3.00 Computer Use: Web and Database Systems

This course offers a practical way of learning the basics of how information is specified, acquired, and managed using database technology. It therefore incorporates four core practices:

- determining the information requirements for a system
- specifying those requirements
- developing a relational database to store the information
- using SQL to manipulate databases

These topics are introduced in a realistic context to promote understanding of how information is used to support business and other organizations. In particular, the course examines the use of database management systems to manage the information content of Web sites. Students also learn to:

- construct web pages in HTML
- design interactive web sites
- design and implement dynamic Web applications

The content for the course is organised in a modular fashion:

1. Introduction to Information Technology and the WWW - introduction to information and database systems, internet information systems (web pages and HTML and web servers)

2. Designing and Specifying Information Systems - data models, entity-relationship diagrams
3. Designing and Creating Relational Databases - developing relational models, defining relational databases in MS Access, improving designs
4. Manipulating Relational Information - using MS Access, using SQL
5. Creating Interactive Web Sites- presenting information with HTML, introduction to ASP and JavaScript, database applications for the web.

Prerequisites: None
Course Credit Exclusions: SB/OMIS 3730 3.00
NCR Note: No credit will be retained if this course is taken after the successful completion of, or simultaneously with EECS3421 3.00 or ITEC3220 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. Through a combination of lectures and laboratory sessions, students become familiar with a scientific computing environment that combines numeric and symbolic computation, high-level programming, scientific libraries, graphics, and a variety of visualization tools. Topics include:

1. Working with the Environment - opening MAPLE, saving your program, getting help
2. Basic Aspects of Maple - MAPLE as a programming language, variables, constants, expressions and assignments, lists, sets, arrays
3. Control Structures - looping, repetition, branching
4. Procedures - defining, calling, parameters and local variables, library procedures, loading a package (example: Linear Algebra), user-defined procedures
5. Data Structures - expressions and operands, strings, lists, arrays and tables
6. Plots and other Visualization Tools - the MAPLE plotting package, plotting tabular data, approximating curves and surfaces, the GRID and MESH objects, animations: Display and Animate commands, generating reports, converting MAPLE plots into images (such as gif, jpeg) and incorporating them into web pages, putting MAPLE Notebooks on the Web
7. Recursion - recurrence relations, reduction formulas from integration, sorting (bubble sort, quick sort), calculation of numbers with recursive formulas
8. Math and Statistics Applications, including possibly topics from: Algebra, Calculus, Probability and Statistics, Matrix Algebra, Trigonometry. Such topics may include, but are not limited to: (Algebra) solve equations and systems of equations, simplify expressions, find roots of polynomials; (Calculus) calculate limits, find derivatives, compute finite sums, evaluate integrals; (Probability and Statistics) Generate and animate: Poisson distribution, exponential distribution, normal distribution, Pseudo-randomly generated data from distributions; (Matrix Algebra) Vectors, operations involving vectors (difference, dot product, cross product), Matrices, operations involving matrices (define a matrix, build a matrix from column vectors and from row vectors, add two matrices, multiply two

matrices, add, multiply, divide entries of a matrix by a value, transposition, determinants); (Trigonometry) MAPLE's trigonometric functions and examples.

Prerequisites: AS/SC/AK/MATH 1300 3.00
Co requisites: AS/SC/AK/MATH 1310 3.00; AS/SC/AK/MATH 1131 3.00
Course Credit Exclusions: EECS 1541 3.00, EECS 1570 3.00
NCR Note: No credit will be retained if this course is taken after the successful completion of, or simultaneously with SC/PHYS 2030 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.

- General introduction to computing and software development, command window, editor, creating and running simple scripts.
- Variables and mathematical operations
- Selection control structures, logical operators, etc.
- Iteration control structures
- File I/O, recording user responses, etc.
- Data types: cell and structure
- Functions
- Plotting
- Creating 2-D graphics
- Simple animation
- MedialLab and DirectRT, SuperLab (use demo version full except for data collection), E-Prime (demo version, full except for data collection)

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: MATH1505 6.00
Course Credit Exclusions: EECS1541 3.00, EECS1560 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). Three lecture hours and weekly 90minute-long laboratory sessions. The laboratory sessions form an integral part of the lectures and may cover examinable material that is not covered in class.

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.

Topics include:

- Digital Media: Introduction and Core Concepts
- Examples of Digital Media artefacts, the notion of evaluation (e.g., the evaluation of software), projects and questions positioned at the intersection of Science and Art
- Why do we use the programming language and environment? (and not Macromedia Director or other tools)
- The use of APIs and other sources of documentation
- Variables and Control Structures
- Iteration
- Modularity (functions, procedures)
- Object-Oriented Constructs (what is a class vs. what is an instance, instantiation, attribute access and method invocation, constructors, encapsulation)
- Integration of Sound, Video (the use of cameras, microphones, other peripherals)
- Application invocation within a networked context (HTTP connections, URLs, sharing information, server file access (read/write))
- The connection between programming languages such as Processing and Java, and other tools for implementation Macromedia Director, Max/MSP, and other Digital Media tools

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

Topics include:

- User Interfaces (UIs)
- UI Elements
- Event driven programming
- Intro to threads
- User Interface Builders
- Guidelines for UI design
- Objects, classes and inheritance
- Interactive WWW-based systems - introduction to WWW and basic network concepts, HTML, Javascript, other WWW technologies (e.g. Flash), guidelines for WWW design
- How to design simple games and make them engaging

Prerequisites: EECS1710 3.00
Course Credit Exclusions: EECS1020 3.00, ITEC1620 3.00, ITEC1630 3.00

Course Descriptions: 2000-Level

General Prerequisites for 2000-level courses is defined as

- EECS1030 3.00 with a grade of C+ or better

Specific prerequisites may also apply to individual courses. Normally a maximum of three EECS courses may be taken in any one of the fall or winter terms (two during the summer term) at any level higher than 1000 provided that prerequisites are met.

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 typically include the following.

- Finite automata and regular expressions; practical applications, e.g., text editors
- Pushdown automata and context-free grammars; practical applications, e.g., parsing and compilers
- Turing machines as a general model of computers; introduction to unsolvability: the halting problem

Prerequisites: General prerequisites; EECS1019 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.

Topics covered may include the following.

- Review of primitive data types and abstract data type — arrays, stacks, queues and lists
- Searching and sorting; a mixture of review and new algorithms
- Priority queues
- Trees: threaded, balanced (AVL-, 2-3-, and/or B-trees), tries
- Graphs: representations; transitive closure; graph traversals; spanning trees; minimum path; flow problems

Prerequisites: General prerequisites; EECS1019 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.

The lab sessions (3 hours/week) involve experiments on assembly and machine language, hardware description languages and simulators, processor architectures, cache memories.

Suggested reading:

- Computer Organization and Design: The Hardware / Software Interface, 3rd edition by D. Patterson and J. Hennessy, Morgan Kaufmann Publishers (2005).
- Structured Computer Organization, 5th edition, by Andrew S. Tanenbaum, Prentice Hall (2006).
- Computer Organization and Architecture: Designing for Performance, 7th edition, by William Stallings, Prentice Hall (2006).

Prerequisites: General prerequisites

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 the following topics:

- ANSI-C (stdio, pointers, memory management, overview of ANSI-C libraries)
- Shell programming

- Filters and pipes (shell redirection, grep, sort & uniq, tr, sed, awk, pipes in C)
- Version control systems and the "make" mechanism
- Debugging and testing
- All the above tools will be applied in practical programming assignments and/or small-group projects.

The course is structured as two hours of weekly lectures and two hours of weekly labs.

Suggested reading:
- Kernighan and Ritchie, The C Programming Language (ANSI C Edition).
- Kernighan and Pike, The Practice of Programming.

Prerequisites: General prerequisites

EECS 2041 4.00 Net-Centric Computing

Net-centric computing encompasses numerous technologies but is based on a few underlying principles. This course covers these principles in general and examines a representative subset of the prevailing technologies. Topics include network programming; web applications; database connectivity; content representation and presentation; and client-side programming.

Detailed topics list:

- Network programming
  - Overview of the Protocol Stack
  - Creating sockets
  - The HTTP (hypertext transfer protocol) standard
  - Multi-tier architectures
- Web programming
  - The CGI (common gateway interface) protocol
  - Server-side scripting, e.g. Perl, PHP, Ruby, Python
  - XHTML and Forms
  - Session management
  - Database connectivity
- Content representation and presentation
  - XML (extensible markup language) and XML Schemas
  - XML Parsing and DOM (document object model)
  - XSL (extensible stylesheet language) and XPATH
  - Content Presentation via XHTML and CSS (cascading stylesheets)
- Client-side programming
  - The scripting engine and JavaScript
  - Host objects, DOM, and events
  - Building rich internet apps via AJAX (asynchronous JavaScript and XML)
  - JavaScript libraries such as JQuery and Dojo

- The Future of the Web
  - Service-Oriented Computing
  - Web Modelling

Security related issues, such as packet sniffing, denial of service, SQL injection, and phishing, will be covered throughout.

Expected Learning Outcomes:
At the end of this course the student will be expected to demonstrate abilities to:

- Build applications out of pieces running on different platforms and communicating through sessions
- Recognize the separate concerns of content generation, transfer, and presentation, and identify the technologies appropriate for each.
- Embed rich interactive applications in a universal client.

Prerequisites: General prerequisites
Course Credit Exclusions: ITEC3020 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. Three lecture hours per week, three laboratory hours every other week.

- The topics will include:
- Introduction, circuit elements, current and voltage sources, power.
- Kirchhoff's laws, dependent sources.
- Nodal analysis, mesh analysis.
- Introduction to 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.

- Magnetically coupled circuits and transformers.

Prerequisites: SC/PHYS 1010 6.00.
Course credit exclusion: SC/PHYS 3050 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. Diodes, transistors (both BJT and FET), amplifiers, rectifiers, frequency response, feedback amplifiers, switching circuits and introduction to VLSI. Three lecture hours per week, three-hour biweekly laboratory.

Topics:

- Diodes: review of 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.
- Differential and multistage amplifiers: MOS differential pair, frequency response, BJT differential pair, multistage amplifiers, operational amplifiers and circuits.
- Switching circuits: Transistors as switches, transistor circuits for logic gates, noise margin, CMOS, fan-in, fan-out, power and speed considerations.
- Introduction to VLSI.

Prerequisites: EECS2200 3.00.
Course credit exclusion: SC/PHYS 3150 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).

After successful completion of the course, students are expected to be able to:

- Understand the many difficulties in developing large-scale software and the need for structured approaches to many issues, such as requirement elicitation, software specification and design, and testing.
- Produce a complete system that matches its requirements and provides a friendly user experience.
- Use several software engineering tools, such as IDEs, source control systems, and configuration management tools.
- Derive and develop effective unit test cases.
- Create user documentation that is accurate and complete.

Prerequisites: General prerequisites

EECS 2501 1.00 Fortran and Scientific Computing

Covers computer-based problem solving in a variety of scientific and engineering settings. 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 the following topics.

- Data types, control structures and program structure
- Functions and subroutines
- Arrays
- I/O
- 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

Prerequisites: EECS1020 3.00 or EECS1530 3.00

EECS 2550 1.00 Introduction to C# Programming (discontinued)

EECS 2560 1.00 C# Programming Tools for Graphical User Interfaces (discontinued)

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. The topics covered in the course will be selected from the following list.

List of Topics:

- Review of complex numbers, trigonometry, and functions.
- Introduction to CT Signals and Systems.
- Properties of CT Systems.
- Representations for Linear, Time Invariant Systems: Differential
- Equations; Convolution Integral; Laplace/Fourier Transfer functions.
- CT Fourier Series for Periodic Signals
- CT Fourier Transform for CT Aperiodic Signals.
- Design of CT Filters: Butterworth, Chebyshev, and Elliptic Filters.
- Applications of CT Systems in Communications and Control.

Learning Objectives for the course:

By the end of the course, the students will be able to:

- 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.

- Analyze 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 analyze and design the CT signals/ systems for selected real- world applications.

The weight distribution of the graded course components is as follows:

- 12% - Assignments (biweekly)
- 12% - Quizzes (following the assignments)
- 24% - Lab Projects (biweekly)
- 20% - Midterm (in-class)
- 32% - Final Exam (scheduled by the Registrar office)

Prerequisites: MATH1014 3.00, MATH1025 3.00
Course Credit Exclusions: EECS3451 4.00

Course Descriptions: 3000-Level

General Prerequisites for 3000-level courses is defined as:

- EECS2011 3.00
- A cumulative grade point average of 4.5 or better over all completed[4] major[5] computer science courses

In computing the GPA the September 2004 Senate legislation applies: If a course is completed more than once, only the second grade is used, unless otherwise directed by the student's home Faculty upon successful petition by the student.

Specific additional prerequisites may also apply to individual courses. A comma or semicolon in a prerequisite list is to be read as "*and*".

Notes:

- Normally a maximum of *three* EECS courses may be taken in any one of the fall or winter terms (two during the summer term) at any level higher than 1000 provided that prerequisites are met.
- *Although Java is used in introductory courses, some upper level courses assume students have a working knowledge of C++, and/or the C programming language; therefore students may want to plan on completing EECS2031 3.00 before entering third year.*
- General prerequisites do not apply to EECS3121 3.00, 3122 3.00 and 3482 3.00.
- The EECS2011 3.00 requirement does not apply to EECS3201 4.00, 3215 4.00, and 3451 4.00 but the above gpa requirement does.

---

[4] *"**Completed**" means that the course appears on your transcript, whether **passed** or **failed**, and is not flagged **NCR** (No Credit Retained).*

[5] *"**Major**" EECS courses are (1) EECS/MATH1019 and (2) all other EECS courses whose second digit is not 5.*

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.

Prerequisites: General prerequisites
Course Credit Exclusions: EATS 3001 1.00, PHYS 3001 1.00, CSE3001 1.00

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.

Topics covered may include the following:

- Review: fundamental data structures, asymptotic notation, solving recurrences
- Sorting and order statistics: heapsort and priority queues, randomised quicksort and its average case analysis, decision tree lower bounds, linear-time selection
- Divide-and-conquer: binary search, quicksort, mergesort, polynomial multiplication, arithmetic with large numbers
- Dynamic Programming: matrix chain product, scheduling, knapsack problems, longest common subsequence, some graph algorithms
- Greedy methods: activity selection, some graph algorithms
- Amortization: the accounting method, e.g., in Graham's Scan convex hull algorithm
- Graph algorithms: depth-first search, breadth-first search, biconnectivity and strong connectivity, topological sort, minimum spanning trees, shortest paths
- Theory of NP-completeness

Suggested reading:

- T.H. Cormen, C.E. Leiserson, R.L. Rivest, *Introduction to Algorithms*, 2nd edition, McGraw-Hill and The MIT Press, 2001.
- J. Edmonds, *How to Think About Algorithms*, Cambridge University Press, 2008.

Prerequisites: General prerequisites; MATH1090 3.00, 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.

Topics covered may include the following:

- Preliminaries—linear algebra, computer programming and mathematical software
- Number systems and errors—machine representation of numbers, floating-point arithmetic, simple error analysis, ill-conditioned problems and unstable algorithms
- Solution of systems of linear equations—Gaussian elimination and its computational complexity, pivoting and stability, special structures (Cholesky's factorization for positive definite systems, banded systems, storage and computational complexities) error analysis, condition number and iterative refinement
- Solution of over determined systems of linear equations by linear least squares approximations—linear least squares problems, normal equations, orthogonal transformations (Given's and Householder's), QR and singular value decompositions (SVD), SVD and rank-deficient problems, computational complexities versus robustness
- Interpolation—Newton's divided differences spline interpolation; banded linear systems, error analysis for interpolation. Other interpolations (rational, B-splines)

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

The final grade will be based on assignments, tests and a final examination.

Prerequisite: EECS3121 3.00

EECS 3201 4.00 Digital Logic Design

Theory and design of logic circuits used in digital systems. This is an intermediate level course that uses a Hardware Design Language to illustrate modern design techniques and is supplemented by hardware laboratory exercises (2 hours per week).

The topics covered will include:

- Review of number systems, Boolean algebra, logic gates and their electrical characteristics.
- Analysis and design of Combinational Circuits including arithmetic units, multiplexers, data selectors, parity checkers etc.
- Hardware Description Languages (HDL). Use of VHDL in logic circuit design and simulation.
- Analysis and design of Sequential Circuits. Flip flops, synchronous and asynchronous circuits. Design using Algorithmic State Machines.
- Memory systems, programmable logic and their applications. Register transfer techniques, Bus concepts.
- Design examples.

Recommended Texts:

- M. Morris Mano, *Digital Design*, (Third Edition), Prentice Hall, 2002.
- S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with VHDL Design*, McGraw Hill, 2001.
- R.S. Sandige, *Digital Design Essentials*, Prentice Hall.

Prerequisites: A cumulative grade point average of at least 4.5 over all completed major computer science courses; EECS2021 4.00. PHYS3150 3.00 is strongly recommended

EECS 3213 3.00 Communication Networks

This course is an introduction to communications and networking. Topics covered include:

- Distinction between information and data, between signal and data, between symbol and data, and between analogue and digital data
- Transmission media; time domain and frequency domain
- Fundamental limits due to Shannon and Nyquist
- Protocol hierarchies; the OSI model
- Encoding of analogue/digital data as analogue/digital signals
- Data link protocols; error and flow control
- Medium access; Ethernet and token passing systems in LANs
- Routing of packets in networks, congestion control

- Internetworking
- Transport services and protocols
- High-level applications and their protocols, e.g. WWW(HTTP), e-mail (SMTP), Internet names (DNS)

Prerequisites: General prerequisites; MATH1310 3.00
Course Credit Exclusions: COSC3211 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 may include:

- Networking Basics
- Queuing Fundamentals
- Network Layer Protocols, Including ICMP, DHCP, and ARP Multicasting
- Transport Layer, UDP, and TCP
- Sockets and Socket Programming
- Application Layer Protocols, Including HTTP and DNS
- Multimedia
- Security
- VOIP

Prerequisites: General prerequisites
Course Credit Exclusions: 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. The following is a detailed list of topics to be covered:

- Introduction to 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 field programmable gate arrays

References:

- M.D. Ciletti, Modeling, Synthesis, and Rapid Prototyping with the VERILOG (TM) HDL, 1st ed, (Prentice-Hall).
- J.K. Peckol, Embedded Systems: A contemporary Design Tool (Wiley).

- W. Wolf, Computers as Components (Morgan-Kaufman).
- F. Vahid and T. Givargis, Embedded System Design: A Unified Hardware/Software Introduction (John Wiley& Sons).

Prerequisites: A cumulative grade point average of at least 4.5 over all completed major computer science courses; EECS2031 3.00; 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 be able to understand the principles and techniques required for understanding and designing operating systems.

Prerequisites: General prerequisites; EECS2021 4.00, EECS2031 3.00
Course Credit Exclusions: COSC3321 3.00

EECS 3301 3.00 Programming Language Fundamentals

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.

Topics covered may include the following:

- Classification of programming languages: language levels, language generations, and language paradigms
- Programming language specification: lexical, syntactic, and semantic levels of language definition
- Data, data types, and type systems: simple types, structured types, type composition rules
- Control primitives, control structures, control composition rules
- Subprograms: functions and procedures, argument-parameter binding, overloading
- Global program structure: modules, generic units, tasks, and exceptions
- Object-oriented language features: classes, encapsulation, inheritance, and polymorphism
- Critical and comparative evaluation of programming languages

Prerequisites: General prerequisites; 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.

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.

Specific topics to be discussed may include the following:

- software design principles: coupling and cohesion, information hiding, open-closed, interface design
- abstract data types
- seamless software construction and process models; a rational design process
- design-by-contract and its implementation in programming languages and design methods; writing and testing contracts; debugging contracts
- abstraction and data design; choosing data structures
- the Business Object Notation (BON) for modelling designs; alternative modelling languages like UML, data-flow diagrams, structure charts, etc.
- static software modelling; dynamic modelling and behavioural modelling
- case studies in design: designing architectures; comparisons; design of OO inheritance hierarchies; class library design
- methods for finding classes; designing class interfaces
- CASE tools: forward and reverse engineering of code from models
- software testing
- design patterns; applications of patterns; implementing patterns

Prerequisites: General prerequisites; EECS2031 3.00, MATH1090 3.00

EECS 3341 3.00 Introduction to Program Verification (discontinued)

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.

After successful completion of the course, students are expected to be able to:

- Understand the nature of formal methods and evaluate their suitability.
- Understand user requirements documents and the distinction between environmental constraints as opposed to functional and safety descriptions.
- Construct high level, abstract mathematical models of a system (consisting of both the system and its environment) amenable to formal reasoning.
- Use 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.
- Understand how to use models to reason about and predict their safety and liveness properties.
- Construct a sequence of refinements from abstract high-level specifications to implemented code and the proof obligations for showing that a concrete system refines the abstract system.
- Know the theory underlying state-event systems, refinements and their proof obligations.
- 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.
- Compare the theory with classical Hoare Logic and the Dijkstra weakest precondition calculus.

Chapters 1, 2, 3, 4, 5 and 9 from the text *Modeling in Event-B: System and Software Engineering*, Jean-Raymond Abrial (Cambridge, to appear 2010) makes this a possible textbook for the course.

Prerequisites: General prerequisites; 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.

The course covers the following topics:

- Introduction to Artificial Intelligence, intelligent agents.

- Logical representations, first-order logic syntax and semantics, use in knowledge representation.
- Basics of logic programming and Prolog, syntax, backchaining procedure.
- Inference in first order logic, unification, resolution.
- Reasoning with Horn theories, SLDNF resolution, Prolog control flow, backtracking, closed world assumption, negation as failure.
- Prolog lists, arithmetic.
- Uninformed search.
- Informed search.
- Constraint satisfaction and backtracking search.
- Game/adversarial search.
- Uncertain reasoning, Bayes Nets.

Prerequisites: General prerequisites; MATH1090 3.00
Course Credit Exclusions: EECS3402 3.00

EECS 3403 3.00 Platform Computing

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). Topics include:

- Introduction to .NET - the .NET Framework, the Common Language Runtime , the Common Type System, common Language Specification, the .NET Framework Class Library, Visual Studio
- NET Languages - C#: Examples, types, non-object-oriented features, object-oriented features; Visual Basic: Examples, types, control structures, non-object-oriented features, object-oriented features.
- .NET Framework Class Library highlights - System namespace, System.IO namespace, System.Collections, System.XML, System.Net, System.Sockets, System.Web, System.Windows.Forms.
- Building Web applications with ASP.NET - .aspx files, web controls, code-behind, etc
- Building Distributed applications (Web Services)
- accessing databases with ADO.NET
- .NET security

Prerequisites: General prerequisites

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 may include the following:

- Overview of Database Management Systems
- Relational Model
- Entity-Relational Model and Database Design
- SQL
- Integrity Constraints
- Crash Recovery
- Concurrency Control

Prerequisites: General prerequisites
Course Credit Exclusions: AK/COSC3503 3.00, 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.

Prerequisites: General prerequisites; EECS2031 3.00, MATH1025 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 lab will cover applications of the concepts covered in the lectures to practical problems such as speech and image processing.

The following topic will be covered

- Continuous and discrete time signals
- Linear time-invariant systems
- Fourier analysis in continuous time
- Fourier analysis in discrete time
- Sampling
- Laplace transform
- Z transform
- Linear feedback systems
- Design of Continuous and discrete time frequency selective filters.

There are three supervised lab hours per week.

Prerequisites: A cumulative grade point average of at least 4.5 over all completed major computer science courses; SC/EECS 2021 4.00; SC/MATH 1310 3.00
Course Credit Exclusions: COSC4242 3.00, COSC4451 3.00, EATS4020 3.00, MATH4130B 3.00, MATH4830 3.00, 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 user interface.

The first part of the course concentrates on the technical aspects of user interfaces (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 user interfaces. 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 user interface evaluation.

Prerequisites: General prerequisites
Course Credit Exclusions: ITEC3230 3.00, ITEC3461 3.00
NCR Note: No credit will be retained by students who successfully completed AS/SC/COSC4341 3.00

EECS 3481 3.00 Applied Cryptography

This course provides an overview of cryptographic algorithms and the main cryptosystems in use today. The course emphasises the application of cryptographic algorithms to designing secure protocols. Topics include:

- Cryptography and Information Security—terminology, information integrity, confidentiality, authentication, non-repudiation.
- Symmetric Key Cryptography - classical ciphers, encryption standards, and modern symmetric ciphers
- Public Key Infrastructure - asymmetric cryptography, hash functions, certificate authorities, digital signatures, cryptanalysis
- Cryptographic Protocols—IP, transport and application layer security (SSL, IPSec, VPN), authentication protocols (Kerberos, single sign-on, biometrics), electronic mail (PGP, S/MIME), XML and WS Security, wireless and broadband security, cryptanalysis 101.
- How can secure systems (that use crypto) be broken?
- Trusted computing—hardware and software aspects.

Prerequisites: General prerequisites

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).

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

The topics that this course covers are the following:

- Foundational concepts of computer security including goals and terminology
- Security Domains (e.g. physical, network, operating system, application etc.)
- Overview of Cryptography
- Security Policies
- Organizational approaches to computer security including different types of security personnel and corresponding certifications
- Laws and Ethics as they relate to computer security
- Risk Management
- Security Auditing (planning, fieldwork, reporting, and follow-up) including professional audit standards

Upon completing this course the student will have demonstrated the ability to do the following (divided by topic):

Foundational Concepts

- Define the meaning of the terms: vulnerability, threat, attack, measure; and give an example of each.
- Describe the three C.I.A. goals of information security and provide an example of an attack against each.
- Explain the difference between passive and active attacks and classify the following attacks accordingly: packet sniffing, IP spoofing, and phishing.
- Provide examples that illustrate the meaning of the following attacks: denial of service, traffic analysis, masquerade, replay, repudiation.

Security Domains

- Give three examples of physical security and provide, for each, an attack example and a counter measure.
- Define the term "social engineering" and provide three examples of attacks that use it.
- Compare and contrast the following malware: virus, worm, botnet, and trojan.

- Explain the difference between operating system security, application security, network security, and web security, and provide, for each, an attack example.

Cryptography

- Define the meaning of the terms: encryption, decryption, ciphertext, plaintext, and key.
- Explain the difference between symmetric and asymmetric cryptography and name two algorithms in each category.
- Describe the notion of a message digest and name two popular algorithms that compute it.
- Show how cryptography can be used to create digital signatures and how these can be used to establish identities.

Security Policies

- Describe the general use of (security) policy and why it has such a central role in a successful information security program.
- Explain the difference between security policy, standard and procedure.
- List different types of security policies that can be found in an organization, and describe what goes into each.
- Develop, implement and maintain various types of information security policy.

Personnel

- Describe different types of information security positions, as well as identify skills and knowledge required for each of the positions.
- List and describe different organization/structural approaches to information security.
- Explain the importance of professional (security) certification, and list the skills, advantages and obligations that are encompassed by each.
- Have knowledge of some useful security practices related to the process of employee hire, termination and misuse control

Laws / Ethics

- Describe the difference between law and ethics, and the importance of each.
- Explain the difference between Criminal and Civil Law, and when an (computer- and/or network- related) offence will be prosecuted under one vs. the other law.
- List major national and international laws that relate to the practice of information security, providing relevant practical examples/cases for each.
- Identify the major professional organizations related to the field of information security, and have general knowledge of their respective Codes of Ethics.

Risk Management

- Define risk management and its role in an organization - both in general and in more specific security-related terms.

- Use risk management techniques to identify and prioritize information assets.
- Assess risk to information assets based on the likelihood of adverse events, and estimate the ultimate effects of the adverse event on information assets.
- Document the results of risk identification and evaluation process.

Auditing

- Explain each phase of a standard Audit from Planning, Fieldwork, Reporting, and Follow-up.
- Describe the purpose of each deliverable in each phase in the Audit.
- Identify and prepare audit planning documents detailing the scope and objectives of the audit.
- Obtain and document sufficient, reliable and relevant evidence to achieve audit objectives, support findings and conclusions as per industry standards.
- Have knowledge of professional audit standards and basic understanding of frameworks.

Students will also have gained practical experience with a variety of security tools including:

- Penetration testing tools, such as the Metasploit framework
- Password crackers, such as John the Ripper or ophcrack
- Vulnerability scanners, such as OpenVAS and Nikto
- Intrusion detection systems, such as snort
- Malware, such as various trojans and viruses

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

EECS 3900 0.00 Computer Science Internship Work Term

This experiential education course reflects the work term component of the Technology Internship Program (TIP). 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 Career Centre 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 Career Centre; 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.

Evaluation:
Performance in each term (EECS3900 0.00) will be graded on a pass/fail basis. To receive a passing grade, the student must pass each of the required components. Note that not all components are required for each Internship term if the Placement consists of more than two terms.

These components are:

- *Employer Evaluation*. Completed by the employer, this summarises the performance of the student at the placement. If the student is engaged in a 12 or 16-month work term placement at the same company, only two evaluations are required. These are due in the second and final term of the placement. The employer evaluation will be submitted to the Internship Coordinator.
- *Internship Coordinator Evaluation*. Completed by the Internship Coordinator, this report is completed based on a minimum of two meetings, at least one normally conducted at the work site. The first one will be conducted at the work site within the first term, and the second as a follow-up either on-site or by telephone or email.
- *Work Report*. Submitted by the student upon his/her return to campus to the faculty supervisor at the end of every work term. This is a short (3-5 page) summary of the work performed during the internship and an assessment of the value of the opportunity. The supervisor will grade the work report and forward it to the Internship Coordinator.

The faculty supervisor assigns the course grade based upon the Employer Evaluation, Internship Coordinator Evaluation, and Work Report.

ENG 3900 0.00 Engineering Internship Term

This experiential education course reflects the work term component of the Technology Internship Program (TIP). 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 Career Centre 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 core engineering credits at the 3000-level including LE/ENG 3000 3.00 (Prior to Summer 2013: SC/ENG 3000 3.00), with a GPA of at least 5.00 in all core engineering courses; 2. that students are enrolled full-time in the

BEng (formerly BASc) degree program prior to beginning their internship and have attended the mandatory preparatory sessions as outlined by the Career Centre; 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 LE/ENG 3900 0.00 (Prior to Summer 2013: SC/ENG 3900 0.00) provides a record on the transcript for each work term.

Evaluation:
Performance in each term (EECS3900 0.00) will be graded on a pass/fail basis. To receive a passing grade, the student must pass each of the required components. Note that not all components are required for each Internship term if the Placement consists of more than two terms.

These components are:

- *Employer Evaluation*. Completed by the employer, this summarises the performance of the student at the placement. If the student is engaged in a 12 or 16-month work term placement at the same company, only two evaluations are required. These are due in the second and final term of the placement. The employer evaluation will be submitted to the Internship Coordinator.
- *Internship Coordinator Evaluation*. Completed by the Internship Coordinator, this report is completed based on a minimum of two meetings, at least one normally conducted at the work site. The first one will be conducted at the work site within the first term, and the second as a follow-up either on-site or by telephone or email.
- *Work Report*. Submitted by the student upon his/her return to campus to the faculty supervisor at the end of every work term. This is a short (3-5 page) summary of the work performed during the internship and an assessment of the value of the opportunity. The supervisor will grade the work report and forward it to the Internship Coordinator.

The faculty supervisor assigns the course grade based upon the Employer Evaluation, Internship Coordinator Evaluation, and Work Report.

EECS 3980 0.00 Computer Security Internship Work Term

This experiential education course reflects the work term component of the Technology Internship Program (TIP). 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 Career Centre 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 Career Centre; 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.

Evaluation:
Performance in each term (EECS3980 0.00) will be graded on a pass/fail basis. To receive a passing grade, the student must pass each of the required components. Note that not all components are required for each Internship term if the Placement consists of more than two terms.

These components are:

- *Employer Evaluation*. Completed by the employer, this summarises the performance of the student at the placement. If the student is engaged in a 12 or 16-month work term placement at the same company, only two evaluations are required. These are due in the second and final term of the placement. The employer evaluation will be submitted to the Internship Coordinator.
- *Internship Coordinator Evaluation*. Completed by the Internship Coordinator, this report is completed based on a minimum of two meetings, at least one normally conducted at the work site. The first one will be conducted at the work site within the first term, and the second as a follow-up either on-site or by telephone or email.
- *Work Report*. Submitted by the student upon his/her return to campus to the faculty supervisor at the end of every work term. This is a short (3-5 page) summary of the work performed during the internship and an assessment of the value of the opportunity. The supervisor will grade the work report and forward it to the Internship Coordinator.

The faculty supervisor assigns the course grade based upon the Employer Evaluation, Internship Coordinator Evaluation, and Work Report.

Course Descriptions: 4000-Level

General Prerequisites for 4000-level courses is defined as:

- EECS2011 3.00

- A cumulative grade point average of 4.5 or better over all completed[6] major[7] computer science courses

In computing the GPA the September 2004 Senate legislation applies: If a course is completed more than once, only the second grade is used, unless otherwise directed by the student's home Faculty.

General prerequisites do not apply to EECS4161 3.00 and 4482 3.00.

Specific additional prerequisites may also apply to individual courses. A *comma* or *semicolon* in a prerequisite list is to be read as "*and*".

Note: Normally a maximum of three EECS courses may be taken in any one of the fall or winter terms (two in the summer term) at any level higher than 1000 provided that prerequisites are met.

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. Internship students may apply to receive credit for their internship as a project course. A "contract" including the seminar presentation is still required.

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

---

[6] *See the definition of "completed" where the general prerequisites of 3000 level courses are listed.*

[7] *See the definition of "major" where the general prerequisites of 3000 level courses are listed.*

EECS 4081 6.00 Intelligent Systems Project

This is an honours thesis course in Intelligent Systems. 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 a single student (or group of students). After two organizational meetings in September, the student will work with his/her 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 student 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 Intelligent System 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 Intelligent Systems Stream will be invited to attend this presentation. The individual supervisor, the course coordinator and one more faculty member will mark this presentation. The final report will be due at the end of the term and will be marked by the individual supervisor.

The actual nature of the project will vary from student to student. Although projects that involve significant implementation are anticipated, purely theoretical projects are possible as well.

Marking Scheme:

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

Prerequisites: Only open to students in the Intelligent Systems Stream who have completed EECS3401 3.00 with a minimum grade of B, and have prior permission of the instructor.

Course Credit Exclusions: EECS 4080 3.00; EECS 4082 6.00; EECS 4084 6.00, EECS 4088 6.00, EECS 4480 3.00, ENG4000 6.00

EECS 4082 6.00 Interactive Systems Project

This is an honours thesis course in Interactive Systems. 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 a single student (or group of students). After two organizational meetings in September, the student will work with his/her 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 student 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 Interactive System 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 Interactive Systems Stream will be invited to attend this presentation. The individual supervisor, the course coordinator and one more faculty member will mark this presentation. The final report will be due at the end of the term and will be marked by the individual supervisor.

The actual nature of the project will vary from student to student. Projects will involve the design, implementation and evaluation of an interactive system. While theoretical projects are possible, the expectation is that all projects evaluate the implementation with human participants and include 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. Projects must deal with systems that interact with a human user. This interaction must be a critical component of the system

Marking Scheme:

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

Prerequisites: Only open to students in the Interactive Systems Stream who have passed EECS3311 3.00 and EECS3461 3.00, and have prior permission of the instructor.

Course Credit Exclusions: EECS 4080 3.00; EECS 4081 6.00; EECS 4084 6.00, EECS 4088 6.00, EECS 4480 3.00, ENG4000 6.00

EECS 4084 6.00 Communication Networks Project

This is an honours thesis course in Communication Networks. 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 a single student (or group of students). After two organization meetings in September, the student will work with his/her 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 student in early January. The format of the 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 Communication Networks projects. Normally held between reading week and the third last week of the 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 Communication Networks Stream will be invited to attend the presentation. The individual supervisor, the course coordinator and one more faculty member will mark this presentation. The final report will be due at the end of the term and will be marked by the individual supervisor.

The actual nature of the project will vary from one student to another. Although projects that involve significant implementation are anticipated, purely theoretical or analysis projects are possible as well.

Marking Scheme:

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

Prerequisites: Only open to students in the Communication Networks Stream who have received a grade of at least B in EECS3451 4.00 and EECS3213 3.00, and have prior permission of the instructor.

Course Credit Exclusions: EECS 4080 3.00, EECS 4081 6.00, EECS 4082 6.00, EECS4088 6.00, 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. Internship students may apply to receive credit for their internship as a project course. A “contract” including the final presentation is still required.

Prerequisites: General prerequisites and permission of the course director. Normally restricted to students who have passed 36 credits in Computer Science.

Course Credit Exclusions: EECS4080 3.00, EECS4081 6.00, EECS4082 6.00, EECS4084 6.00, 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.

The goal of this course is to provide 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 ongoing 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.

Prerequisites: Only open to students in the Software Development Stream. B or higher in EECS3311 3.00, and completion of EECS3101 3.00, EECS3342 3.00
Co requisites: EECS 4312 3.00, EECS 4313 3.00
Course Credit Exclusions: none

EECS 4101 3.00 Advanced Data Structures (integrated with CSE5101 3.00)

The course discusses advanced data structures: 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, 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.
The course may include the following topics:

- Amortized and worst-case analysis of data structures
- Data structuring paradigms: self-adjustment and persistence
- Lists: self-adjustment with the move-to-front heuristic
- Search trees: splay trees, finger search trees
- Heaps: skew heaps, Fibonacci heaps
- Union-find trees

- Link-and-cut trees
- Multidimensional data structures and dynamization

Prerequisites: General prerequisites; EECS2001 3.00, EECS3101 3.00.

EECS 4111 3.00 Automata and Computability (integrated with CSE5111 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; EECS2001 3.00, 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 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 diagonalization, 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/Szelepsenyi 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
- If time permits the course may also include one or more advanced topics such as parallel complexity classes, interactive proofs, applications to cryptography, and probabilistic classes including random polynomial time

Possible Text:

- Arora and Barak, Complexity Theory, A modern approach, manuscript, 2008.
- Sipser, M., Introduction to the theory of computation (second edition), Course Technology, 2005.

References:

- C.H. Papadimitriou, Computational Complexity, ISBN: 0-201-53082-1, Addison Wesley, 1994.
- U. Schoning and Randall Pruim, *Gems of Theoretical Computer Science*, ISBN 3-540-64425-3, Springer Verlag, 1998.

- Lane A. Hemaspaandra and Mitsunori Ogihara, *The Complexity Theory Companion,* ISBN 3-540-67419-5, Springer-Verlag, 2002.
- M.R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, ISBN 0716710455, W.H. Freeman, 1979.
- D.-Z. Du and K. Ko, *Theory of Computational Complexity*, ISBN: 0-471-34506-7, John Wiley and Sons, New York, NY, 2000.
- D. P. Bovet and P. Crescenzi, *Introduction to the Theory of Complexity*, ISBN 0139153802, Prentice-Hall, 1993.

Prerequisites: General prerequisites; EECS2001 3.00, 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 analyzing 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 EECS3101 3.00, or permission of the instructor.

EECS 4201 3.00 Computer Architecture (integrated with CSE5501 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 the following:

- Fundamentals of computer design
- Performance and cost

- Instruction set design and measurements of use
- Basic processor implementation techniques
- Pipeline design techniques
- Memory-hierarchy design
- Input-output subsystems
- Future directions

Prerequisites: General prerequisites; EECS3201 4.00, EECS3221 3.00

EECS 4210 3.00 Architecture and Hardware for Digital Signal Processing

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 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 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.

The course includes 6 two-hour lab sessions during which students design special purpose architecture for digital signal processing algorithms using digital signal processor boards and FPGA boards.

Prerequisites: General prerequisites; EECS3201 4.00; EECS3451 3.00

EECS 4211 3.00 Performance Evaluation of Computer Systems
(integrated with CSE5422 3.00)

Topics covered may include the following:

- 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
- Application — A 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; MATH2030 3.00, 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:

- 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

The course includes weekly two-hour lab sessions and a weekly one-hour tutorial.

References:

- Bernard Sklar, Digital Communications: Fundamentals and Applications, NY: Prentice Hall, 2001, 2nd edition, ISBN # 0-13-084788-7 (required).
- John G. Proakis, Digital Communications, Third Edition, McGraw Hill (suggested).
- Simon Haykin, Digital Communications, John Wiley & Sons (suggested).
- Marvin K. Simon, Sami M. Hinedi, and William C. Lindsey, Digital Communication Techniques, NY: Prentice Hall, 1995 (suggested).
- Marvin E. Frerking, Digital Signal Processing in Communication Systems, NY: International Thomson Publishing (ITP), 1994 (suggested).

Prerequisites: General prerequisites; EECS3213 3.00; MATH2030 3.00; one of EECS3451 4.00, EATS 4020 3.00, MATH 4830 3.00, PHYS 4060 3.00, or PHYS 4250 3.00
Course Credit Exclusions: EECS4214 3.00

EECS 4215 3.00 Mobile Communications (integrated with CSE5431 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 the following:

- 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.

The pedagogical components of the course include lectures, office hours, hands-on laboratories and exercises, assignments, tests, and a project that addresses recent research issues in wireless mobile networking.

Two-hour lab sessions will be held alternate weeks. The scheduled lab sessions will involve the use of:

- a commercial software tool for designing and planning of cellular systems (currently EDX);
- a wireless network simulator (currently Qualnet);
- software and hardware tools for building and monitoring of wireless LAN systems (currently the tools from the Cisco wireless family of products).

Prerequisites: General prerequisites; 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.

A solid background in operating systems concepts, computer architecture, C, and UNIX is expected.

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

EECS 4301 3.00 Programming Language Design (integrated with CSE5423 3.00)

This course is a continuation of EECS3301 3.00 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; EECS3301 3.00

EECS 4302 3.00 Compilers and Interpreters (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.

Specific topics to be covered may include the following:

- Compiler architecture: single-pass vs. multiple-pass translation
- Lexical analysis (scanning): design of scanners using finite automata; tabular representations; tools for building scanners
- Parsing (syntax analysis): top-down vs. bottom-up parsing; parse trees and abstract syntax trees; tabular representations for parsers; parser generators
- Symbol tables: efficient algorithms and data structures; representing data types in symbol tables
- Type checking: scope control; static vs. dynamic type checking

- Memory management: static allocation; register allocation; stack allocation; heap allocation; garbage collection
- Code generation: translating imperative programming constructs; function and procedure calls; branching code; translating object-oriented constructs and modules
- Optimization: local and global optimizations; dead code removal; control flow analysis

Prerequisites: General prerequisites; EECS3301 3.00 recommended

EECS 4311 3.00 System Development

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. In particular, JSD is well suited to the following:

- Concurrent software where processes must synchronise with each other
- Real time software. JSD modelling is extremely detailed and focuses on time at the analysis and design stages.
- Microcode. JSD is thorough; it makes no assumptions about the availability of an operating system.
- Programming parallel computers. The JSD paradigm of many processes may be helpful.

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 EECS3311 3.00 or EECS3221 3.00

EECS 4312 3.00 Software Engineering Requirements

This course deals with the elicitation, specification and analysis of software requirements. It provides a critical description of available methods and tools, and practical exercises on applying these methods and tools to realistic problems.

Topics include:

- Requirements and system concepts

- Traceability through requirements into design
- Current requirements methods, techniques, and tools
- Industrial practice and standards
- Specific topics to be covered include:
- Introduction: Problems, principles and processes of requirements engineering
- Requirements elicitation processes and methods
- Introduction to Use Cases and UML
- Specification techniques: Requirements models; data modelling; functional models; the application of formal requirements methods
- Goal-oriented requirements modelling
- Non-functional requirements: safety, security and other non-functional requirements
- Pragmatic requirements engineering: Technology transfer; Traceability
- Current Requirements Standards, e.g., IEE 830 Recommended Practice for Requirements Engineering
- Requirements Categorization for Resource Allocation
- Why-Because Analysis

References:

- G. Kotonya and I. Somerville. Requirements Engineering: Processes and Techniques, Wiley, 1998.
- A. Davis, Software Requirements, Addison-Wesley, 1992.
- S. Robertson and J. Robertson, Mastering the Requirements Process, Addison-Wesley, 1999.
- M. Jackson, Problem Frames, Addison-Wesley, 2000.
- M. Jackson, Software Requirements and Specifications, Addison-Wesley, 1995.

Prerequisites: General prerequisites; 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.

Students should:

- understand the importance of systematic testing
- understand how verification is an integral part of the development process and not a bolt on activity
- understand the strengths and weaknesses of particular techniques and be able to select appropriate ones for a given situation
- 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.

- 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. Theoretical background (e.g., graph theory).
- 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. Specific problems and existing techniques, e.g., Junit, automatic test case generation via UML diagrams.
- 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. Q.A. Controlling the test process. Inspections reviews, walkthroughs and audits. Influence of standards.
- Regression testing.

References:

Primary:

- Paul C. Jorgensen, Software Testing: A Craftsman's Approach, CRC Press, 2002.

Supplementary:

- Robert Binder, Testing Object-Oriented Systems, Addison-Wesley, 2000.
- K. Beck, Test Driven Development By Example, Addison-Wesley, 2002.
- Cem Kaner, James Bach, Bret Pettichord, Lessons learned in software testing : a context-driven approach, Wiley, 2002

Prerequisites: General prerequisites; EECS3311 3.00

EECS 4314 3.00 Advanced Software Engineering (not offered in 2014/15)

This course goes into more detail about some of the software engineering techniques and principles presented in earlier courses, as well as introduces advanced aspects of software engineering that are not addressed elsewhere:

- 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.

After successful completion of the course, students are expected to be able to:

- 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; EECS 3311 3.00

EECS 4315 3.00 Mission-Critical Systems (not offered in 2014/15)

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.

After successful completion of the course, students are expected to be able to:

- 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; EECS3342 3.00

EECS 4351 3.00 Real-Time Systems Theory (integrated with CSE5441 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 the following:

- 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; EECS3221 3.00

EECS 4352 3.00 Real-Time Systems Practice (integrated with CSE5442 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. Twelve supervised laboratory hours (two hours, alternate weeks).

The course will focus 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

Prerequisites: General prerequisites; EECS3221 3.00

EECS 4401 3.00 Artificial Intelligence (integrated with CSE5326 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; EECS3401 3.00

EECS 4402 3.00 Logic Programming (integrated with CSE5311 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.

Topics covered may include the following:

- Logical preliminaries: syntax and semantics of first order predicate logic and its Horn logic fragment
- Logical foundations of logic programming: unification, the resolution rule, SLD-resolution and search trees
- PROLOG as a logic programming system
- Programming techniques and applications of PROLOG

- Constrained logic programming systems

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; EECS3401 3.00; one of EECS3101 3.00 or EECS3342 3.00

EECS 4403 3.00 Soft Computing

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 expectations include (see also section on Course Design for evidence that these expectations will be met):

- 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.

The course will be sectioned into parts:
Part I – Fuzzy Sets and Fuzzy Logic
Part II – Rough Sets
Part III – Neural Networks
Part IV – Evolutionary Computing
Part V – Probabilistic Reasoning
Part VI – Applications, Intelligent Systems design, Hybrid Systems
Part VII – Student Presentations

Parts I-VI will primarily rely on instructor lectures with significant student involvement; Part VII will rely on students making presentations (e.g., their projects).

Prerequisites: General prerequisites; EECS2031 3.00

EECS 4404 3.00 Introduction to Machine Learning and Pattern Recognition
(integrated with CSE5327 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.

Topics may include:

- Introduction to Bayesian decision theory
- Survey of key probability distributions
- Non-parametric modelling
- Mixture models and expectation maximization
- Subspace models
- Linear regression
- Linear models for classification
- Cross-validation
- Kernel methods
- Sparse kernel machines
- Introduction to graphical models
- Bagging & boosting
- Sampling techniques

Learning Objectives:

Upon completing this course the student will, through the assignments, tests, and final project, have demonstrated ability to:

- 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; one of MATH2030 3.00 or 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 may include the following:

- Query Processing
- Transactions
- Concurrency Control
- Recovery
- Database System Architectures
- Distributed Databases
- Object-Oriented Databases

Suggested reading:

- R. Elmasri and S.B. Navathe, *Fundamentals of Database Systems*, 2nd Ed., Benjamin Cummings, 1994.

Prerequisites: General prerequisites; EECS2021 4.00, EECS2031 3.00, 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.

Suggested reading:

- Jiawei Han and Micheline Kamber, Data Mining -- Concepts and Techniques, Morgan Kaufmann, Second Edition, 2006.
- Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Introduction to Data Mining, Addison Wesley, 2006.

- Ian H. Witten and Eibe Frank, Data Mining -- Practical Machine Learning Tools and Techniques (Second Edition), Morgan Kaufmann, 2005.
- Margaret H. Dunham, Data Mining -- Introductory and Advanced Topics, Prentice Hall, 2003.

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

EECS 4413 3.00 Building E-Commerce Systems

A study of technological infrastructure for Electronic Commerce on the Internet discussing terminology, possible architectures, security and cryptography, content presentation, web protocols, adaptive and intelligent agents, data mining, and vertical applications.

Topics covered may include the following:

- Basic e-commerce concepts. Examples of e-commerce stores
- Internet as the infrastructure for e-commerce; network layers and protocols; network and transport layer; TCP/IP; web server design; DNSs, URLs, and HTTP; proxies, caching
- Security and encryption; basic concepts of computer cryptography; symmetric and asymmetric cryptosystems; DES; public key cryptosystems; RSA; Diffie-Hellmann; elliptic codes; PGP; breaking computer cryptography with massive parallelism
- Electronic store content and presentation; HTML, CGI, Dynamic HTML, JavaScript. Applets; push and pull content; MIME and cookies; future representations — XML, WAP
- Intelligent e-commerce; data mining in e-commerce; agents; product and merchant brokerage; mobile agents; negotiations

Prerequisites: General prerequisites

EECS 4421 3.00 Introduction to Robotics (integrated with CSE5324 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.

The course includes 12 hours of supervised lab sessions.

Prerequisites: General prerequisites; MATH1025 3.00, MATH1310 3.00, EECS2031 3.00

EECS 4422 3.00 Computer Vision (integrated with CSE5323 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.

Prerequisites: General prerequisites; MATH1025 3.00; MATH1310 3.00; 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 of the course will include:

1. Molecular biology for computer scientists
   - The cell and the molecules of life: DNA, RNA, chromosomes, genes, transcription, translation, splicing, replication, recombination
   - The Central Dogma of Molecular Biology
   - Proteins: structure and functions
2. Sequence analysis algorithms
   - Scoring matrices
   - Gaps
   - Pairwise global and local alignment: dynamic programming algorithms for general gap penalty and affine gap penalty
   - Multiple global alignment: dynamic programming algorithm and heuristic algorithms
   - Progressive alignment algorithm, CLUSTAL
3. NCBI, National Center for Biotechnology Information
4. BioJava: Java tools for processing biological data
5. Biological databases
   - Databases containing nucleotides and Proteins information: GeneBank, PDB, EST, UniGene, etc. (data formats, methods to connect different databases)
   - Databases containing literature information: PubMed, Public Library of Science
   - Heuristic algorithms for search in biological databases: BLAST, FASTA
   - New algorithms for search in a biological database
6. Phylogenetic trees
   - Algorithms for Reconstruction of Phylogenetic Trees: distance based and character based
   - Algorithms for the Maximum Parsimony and Maximum Likelihood Problems
   - Subtrees and Supertrees: Algorithms
   - Evaluation of Phylogenies Using Bootstrapping
7. Introduction to Microarray Data Analysis for Gene Expression
   - Normalization
   - Pearson correlation
   - Algorithms for Hierarchical Cluster Analysis of Microarray Data
   - An Open Problem: Annotation of Microarray Data

Prerequisites: General prerequisites

EECS 4431 3.00 Advanced Topics in 3D Computer Graphics (integrated with CSE5331 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.

Scheduled lab sessions involve practical experimentation with advanced computer graphics and will support the development, presentation and demonstration of a comprehensive student design project. Two-hour lab sessions will be held during 6 weeks of the course.

Prerequisites: General prerequisites; EECS2021 4.00; EECS3431 3.00
Course Credit Exclusions: COSC4331 3.00

EECS 4441 3.00 Human Computer Interaction (integrated with CSE5351 3.00)

- Introduction (Goals, Motivation, Human Diversity)
- Theory of Human-Computer Interaction (Golden Rules, Basic Principles, Guidelines)
- The Design Process (Methodologies, Scenario Development)
- Expert Reviews, Usability Testing, Surveys and Assessments
- Software Tools (Specification Methods, Interface-Building Tools)
- HCI Techniques
- Interaction Devices (Keyboards, Pointing Devices, Speech Recognition, Displays, Virtual Reality Devices)
- Windows, Menus, Forms and Dialog Boxes
- Command and Natural Languages (Command Line and Natural Language Interfaces)
- Direct Manipulation and Virtual Environments
- Manuals, Help Systems, Tutorials

- Hypermedia and the World Wide Web (Design, Creation, Maintenance of Documents)
- Human Factors—Response Time and Display Rate; Presentation Styles—Balancing Function and Fashion (Layout, Colour); Societal Impact of User Interfaces (Information Overload); Computer Supported Cooperative Work (CSCW, Synchronous and Asynchronous); Information Search and Visualization (Queries, Visualization, Data Mining)

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.

Suggested reading:

- Alan Dix, Janet Finlay, Gregory Abowd, Russell Beale, Human-Computer Interaction, 3rd ed, Prentice Hall, 2004.

Prerequisites: General prerequisites; 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 topics taught in this course include the following:

- Development tools and environment
- Programming and application components for mobile user interfaces
- Touch input
- Location and mobile sensing
- Media, camera, and audio capture
- Text input

The course format is 3 hours of lectures every week and 2 hours of lab exercises every other week.

Prerequisites: General prerequisites; 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. In the second part, it covers some important DSP-based applications in the real world.

The topics to be covered may include:

Part A: DSP theory

Review of discrete-time systems and sampling, review of Z-transforms, discrete Fourier transform (DFT), Fast Fourier transform (FFT); digital filter design - classical filter theory, FIR filters, IIR filters, filter banks, adaptive digital filters, spectral estimation and analysis

Part B: DSP applications (selectively covered by the instructor)

1. Embedded DSP systems: Introduction to DSP processors, architecture and programming, design of embedded DSP systems with TMS320 series
2. Speech and audio processing: Digital waveform coding: PCM, u-law, A-law, Time domain analysis, Short-time spectrum analysis, Linear prediction analysis, Pitch detection and tracking, Speech coding, Music processing
3. Image processing: Two-dimensional signals and systems, Image compression, Image enhancement and restoration, radar and sonar signal processing: array signal processing

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.

The course components include: lectures, assignments, 12 supervised lab hours for 2-3 lab projects, one midterm test, one final exam.

Prerequisites: General prerequisites; EECS3451 3.00

EECS 4461 3.00 Hypermedia and Multimedia Technology

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.

We will explore the following topics.

- The historical roots of hypermedia: Bush, Engelbart, and Nelson
- The digital representation of media: rich text, sound, speech, images, animation, and video
- Enabling technologies for creating hypermedia
- The role of scripting and mark-up languages
- Networked hypermedia (e. g. HTTP browsers); performance and compression issues
- Development Tool Kits
- Distribution and Intellectual Property Issues

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; EECS3461 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 topics will be approximately as follows:

- Introduction: applications, human sensory/motor system & capabilities
- Review of interactive 3D graphics programming. Real-time rendering (levels-of-detail, impostors, etc.), graphics hardware, distributed rendering.
- Virtual Reality Technology (VR): VR input devices, filtering & tracking, VR output devices, Augmented Reality (AR) hardware, spatial audio, haptics
- Virtual Environments (VE): event driven simulation, procedural animation, physics-based modelling, collision detection & response, simulation & rendering in parallel, interaction with VE, haptic and auditory simulation
- Human Factors: presence, immersion, simulator sickness (frame-rate, latency, vergence vs. accommodation, visual vs. vestibular, etc), training (fidelity, transfer)
- Applications: training, collaborative virtual environments, medical, visualization & decision support, design, entertainment, augmented reality, space applications, teleoperation, computer games.

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.

Prerequisites: General prerequisites; MATH1025 3.00; MATH1310 3.00; EECS2021 4.00; EECS2031 3.00; EECS3431 3.00 (may be waived on an individual basis, please consult the instructor).

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.
The actual nature of the project will vary from student to student. However, after successful completion of the course, students are typically expected to be able to:

- 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 EECS credits. Permission of the course director is required.

Course Credit Exclusions: EECS 4080 3.00, EECS 4081 6.00, EECS 4082 6.00, EECS 4084 6.00, EECS4088 6.00, 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.

- 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.

This is a lecture-based course with a laboratory of 3 hours per week.

Prerequisites: General prerequisites; EECS3221 3.00, EECS3214 3.00

Note. Students with background equivalent to the stated prerequisites are encouraged to seek permission to enrol.

EECS 4482 3.00 Computer Security Management: Assessment and Forensics

- 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

Prerequisites: Any 12 credits at the 3000-level

EECS 4491 3.00 Simulation and Animation for Computer Games

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 covered:

- Principles of "Classic" Animation
- Spaces, Transformations, and Rotations
- Interpolation Methods
- Interpolation-Based Animation
- Kinematic Linkages
- Inverse Kinematics
- Motion Capture
- Physically Based Animation
- Liquids & Gases
- Modelling and Animating Human Figures
- Facial Animation
- Modelling Behaviour
- Special Models for Animation

After successful completion of this course, students are expected to be able to understand the concepts behind and to implement:

- 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; EECS3431 3.00, MATH1310 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*: EECS4080 3.00; EECS4081 6.00; EECS4082 6.00; EECS4084 6.00, EECS4088 6.00, EECS4480 3.00

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://www.cse.yorku.ca/undergrad/guides/enroll.html for an expanded description and interpretation of the enrolment policy outlined below.)

Application for Normal Progress

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. However, 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:

- Normally taking 1000-level courses in calendar year one, 2000-level in calendar year two, etc.
- Taking up to a total of three courses per term (four in the cases of engineering majors due to their heavier degree requirements) that are any combination of 2000- 3000- and 4000-level courses 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) 2000- 3000- or 4000-level 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*.

Removal from Courses

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

Prerequisites

Students are *responsible for being aware of the prerequisites of the EECS courses into which they plan to enrol, and for ensuring that they enrol only if* they meet the prerequisites. *Most* prerequisites include a *minimum cumulative GPA (4.5) over all computer science courses taken*. In the course of prerequisite auditing that the Undergraduate Office performs—the process starts at the beginning of each term—students will be removed from a course if they do not meet the prerequisites. *Due to the manual and time consuming process of prerequisite auditing, removal from courses in the case of non-compliance may take place at any time before the start, or*

during the course. *As such, it is imperative that students review the prerequisites of their selected courses*. Students who are de enrolled are notified by email.

Courses taken outside the Department

Students wishing to take Computer Science courses at another institution should submit a Letter of Permission (LOP) form to the Undergraduate Office. 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*.

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, EECS1020, EECS1030, EECS2001, EECS2011, EECS2021, EECS2031, EECS3101, and EECS3311. Core mathematics courses are MATH1300 3.00, MATH1310 3.00, and MATH1090 3.00.

Normal Order of Study

This section presents a summary of course requirements only for the *computer science* programs, by suggesting the normal order in which courses should be taken. *See also under the heading "Limits on Course Enrolment".* Students are strongly encouraged to consult the checklists for each program type (computer science, computer security, digital media and computer engineering) at the end of this calendar (hard copy version) or on-line at the URL
http://www.cse.yorku.ca/cscurrent_students/undergrad_students/index.html

Archived checklists for previous years, and earlier versions of the supplemental calendar are found at the URL http://www.cse.yorku.ca/undergrad/csCalendars.html

The terms "first year", "second year", etc., below indicate the year of study for *normal progress* by *full-time students in a computer science degree*. We note that progress in one's program of study is not based on *year* of study but rather on attainment of the *prerequisites*. Thus the following is an illustration only.

1000-level — first year

- Fall — EECS1001 1.00, EECS1020 3.00, EECS1019 3.00, MATH1300 3.00.
- Winter — EECS1001 1.00 (continued from the fall term – this course meets once every two weeks and spans fall and winter terms) EECS1030 3.00, MATH1310 3.00.
- Additional credits toward satisfying general education, Faculty requirements, second major program, or elective requirements for an approximate grand total of 30 credits.

- Normal progress is one EECS course per term (in the context of this restriction 1019 is viewed as a MATH course, and 1001 does not add to the count, due its small credit weight).

2000-level — second year

- EECS2001 3.00, EECS2011 3.00, EECS2021 4.00, EECS2031 3.00 (and EECS2041 4.00 for all honours programs); MATH1090 3.00.[8]
- *Specialised Honours:* MATH1025 3.00, MATH2030 3.00.
- *Other Honours programs:* MATH2030 3.0.
- Additional credits toward satisfying general education, Faculty requirements, second major program, or elective requirements for an approximate grand total of 30 credits.
- Normal progress is three EECS courses per term.

3000-level — third year

- 12 EECS credits minimum at the 3000-level satisfying the breadth requirement — EECS3101 3.00, EECS3311 3.00, and one of EECS3221 3.00 or EECS3215 4.00.
- 3 credits from among EECS3401 3.00, EECS3421 3.00, EECS3461 3.00.
- *All BA and BSc Honours (120-credit) programs:* EECS3000 3.00.
- *BA and BSc (90-credit) programs*: 6 additional EECS 3000-level credits.
- *BA and BSc Specialised Honours programs:* 3 additional EECS 3000-level credits
- Additional credits toward satisfying general education, Faculty requirements, second major program, or elective requirements for an approximate grand total of 30 credits.
- Normal progress is three EECS course per term.

4000-level — fourth year, honours programs only

- 12 EECS credits at the 4000-level (except for the Honours Minor BA degree which normally requires a maximum of 6 credits at the 4000-level), including one of EECS4101 3.00 or EECS4111 3.00 or EECS4115 3.00 for the Specialised Honours programs.
- 6 additional EECS credits at the 3000- or 4000-level for Specialised Honours programs.
- Additional credits toward satisfying general education, Faculty requirements, second major program, or elective requirements for an approximate grand total of 30 credits.
- Normal progress is three EECS courses per term.

---

[8] *Although MATH1090 is not a 3000-level general prerequisite it is required for some 3000-level core courses and therefore students should plan to complete it in year two.*

Prerequisites for Computer Science Courses[9]

It is required that students fulfil the prerequisites for courses they wish to take.

There are both **general** prerequisites that are required for all EECS courses *at the specified level* and **specific** prerequisites for each course that are in addition to the general prerequisites. Both types of prerequisites include EECS courses and mathematics courses, and in most cases there are grade requirements in the prerequisite courses. The prerequisites are listed after each course description and summarised in the following tables.

The prerequisites table is useful to determine what courses must be taken in order to enrol in a particular course, or to determine if you are permitted to enrol in a course.

Course Title		Prerequisite(s)[10]
1000-Level		
EECS1001 1.00	Research Directions in Computing	See course description
EECS1011 3.00	Computational Thinking through Mechatronics	See course description
EECS/MATH1019 3.00	Discrete Mathematics for Computer Science	MATH1190 3.00, or two 4U courses including MHF4U (Advanced Functions)
EECS1020 3.00	Intro. to Computer Science I	See course description
EECS1021 3.00	Object Oriented Programming from Sensors to Actuators	EECS1011 3.00
EECS/MATH1028 3.00	Discrete Mathematics for Engineers	MHF4U and MCV4U
EECS1030 3.00	Intro. to Computer Science II	EECS1020 3.00 or EECS1720 3.00
EECS1710 3.00	Programming for Digital Media	See course description
EECS1720 3.00	Building Interactive Systems	EECS1710 3.00

2000-Level

[9] *In exceptional circumstances some prerequisites or co requisites may be waived at the discretion of the undergraduate director in consultation with the course director. All petitions to have pre- and co requisites waived must be submitted to the undergraduate office. Course directors may not waive prerequisites.*

[10] *A comma or a semicolon is interpreted as an "and" in a prerequisite list (unless this is overridden by a phrase such as "one of").*

General Prerequisites:

- EECS1030 3.00 completed with a grade of C+ or better

EECS2001 3.00	Intro. to the Theory of Computation	General prerequisites, EECS1019 3.00
EECS2011 3.00	Fundamentals of Data Structures	General prerequisites, EECS1019 3.00
EECS2021 4.00	Computer Organization	General prerequisites
EECS2031 3.00	Software Tools	General prerequisites
EECS2041 4.00	Net-Centric Computing	General prerequisites
EECS2311 3.00	Software Development Project	General prerequisites
EECS2602 4.00	Signals and Systems in Continuous Time	MATH1014 3.00, MATH1025 3.00

3000-Level

General Prerequisites[11]:

- EECS2011 3.00
- A cumulative GPA of 4.5 or better over all completed[12] major Computer Science courses (NB. EECS1019 3.00 is a major EECS course)

Theory and Numerical Computation	Specific Prerequisites
EECS3101 3.00 Design and Analysis of Algorithms	MATH1090 3.00, MATH1310 3.00
EECS3121 3.00 Intro. to Numerical Computations I	One of EECS1540 3.00, EECS2031 3.00, EECS2501 1.00; one of MATH1010 3.00, MATH1310 3.00, or MATH1014 3.00; one of MATH1021 3.00, MATH1025 3.00, or MATH2221 3.00
EECS3122 3.00 Intro. to Numerical Computations II	EECS3121 3.00

Systems

[11] *Applicable to all except EECS3121 3.00, EECS3122 3.00, EECS3201 4.00, EECS3215 4.00, EECS3451 4.00, and EECS3482 3.00 below.*

[12] *"Completed" is defined where the 3000 level and 4000 level general prerequisites are listed.*

EECS3201 4.00 Digital Logic Design	A cumulative grade point average of at least 4.5 over all completed major computer science courses; EECS2021 4.00. PHYS3150 3.00 is strongly recommended
EECS3213 3.00 Communication Networks	MATH1310 3.00
EECS3214 3.00 Computer Network Protocols and Applications	
EECS3215 4.00 Embedded Systems	A cumulative grade point average of at least 4.5 over all completed major computer science courses; EECS2031 3.00, EECS3201 4.0
EECS3221 3.00 Operating System Fundamentals	EECS2021 4.00, EECS2031 3.00

Software Development

EECS3301 3.00 Programming Language Fundamentals	EECS2001 3.00
EECS3311 3.00 Software Design	EECS2001 3.00, EECS2031 3.00, MATH1090 3.00
EECS 3342 3.00 System Specification and Refinement	MATH1090 3.00

Applications

EECS3401 3.00 Introduction to Artificial Intelligence and Logic Programming	MATH1090 3.00
EECS3421 3.00 Introduction to Database Systems	
EECS3431 3.00 Introduction to 3D Computer Graphics	EECS2031 3.00, MATH1025 3.00
EECS3451 4.00 Signals and Systems	A cumulative grade point average of at least 4.5 over all completed major computer science courses; EECS2021 4.00; MATH1310 3.00
EECS3461 3.00 User Interfaces	
EECS3481 3.00 Applied Cryptography	
EECS3482 3.00 Applied Cryptography	Any 12 university credits at the 2000-level in any discipline

Other Courses:

EECS3000 3.00 Professional Practice in Computing	(Required of all EECS honours degrees.)

4000-Level

General Prerequisites[13]:

- EECS2011 3.00
- A cumulative GPA of 4.5 or better over all completed major computer science courses

Theory Courses	Specific Prerequisites
EECS4101 3.00 Advanced Data Structures	EECS2001 3.00, EECS3101 3.00
EECS4111 3.00 Automata and Computability	EECS2001 3.00, EECS3101 3.00
EECS4115 3.00 Computational Complexity	EECS2001 3.00, EECS3101 3.00
EECS4161 3.00 Introduction to Cryptography	At least 12 credits from 2000-level (or higher) MATH courses (without second digit 5); or EECS3101 3.00

Systems Courses	
EECS4201 3.00 Computer Architecture	EECS3201 4.00, EECS3221 3.00
EECS4210 3.00 Architecture and Hardware for Digital Signal Processing	EECS3201 4.00, EECS3451 4.00
EECS4211 3.00 Performance Evaluation of Computer Systems	MATH2030 3.00, EECS3213 3.00
EECS4214 4.00 Digital Communications	EECS3213 3.00, MATH2030 3.00, one of EECS3451 4.00, EATS 4020 3.00, PHYS 4250 3.00
EECS4215 3.00 Mobile Communications	EECS3213 3.00
EECS4221 3.00 Operating System Design	EECS3221 3.00

Software Courses	
EECS4301 3.00 Programming Language Design	EECS3301 3.00
EECS4302 3.00 Compilers and Interpreters	(EECS3301 3.00 recommended)
EECS4311 3.00 System Development	EECS3311 3.00 or EECS3221 3.00
EECS4312 3.00 Software Engineering Requirements	EECS3311 3.00
EECS4313 3.00 Software Engineering Testing	EECS3311 3.00
EECS4314 3.00 Advanced Software	EECS3311 3.00

[13] *Applicable to all except EECS4161 3.00 and EECS4482 3.00 below.*

Engineering	
EECS4315 3.00 Mission-Critical Systems	EECS3342 3.00
EECS4351 3.00 Real-Time Systems Theory	EECS3221 3.00
EECS4352 3.00 Real-Time Systems Practice	EECS3221 3.00

Applications Courses

EECS4401 3.00 Artificial Intelligence	EECS3401 3.00
EECS4402 3.00 Logic Programming	EECS3401 3.00, one of EECS3101 3.00 or EECS3341 3.00
EECS4403 3.00 Soft Computing	EECS2031 3.00
EECS4404 3.00 Introduction to Machine Learning and Pattern Recognition	MATH2030 3.00 or MATH1131 3.00
EECS4411 3.00 Database Management Systems	EECS2021 4.00, EECS2031 3.00, EECS3421 3.00
EECS4412 3.00 Data Mining	EECS3101 3.00, EECS3421 3.00, and one of MATH2030 3.00 or MATH1131 3.00
EECS4413 3.00 Building E-Commerce Systems	
EECS4421 3.00 Introduction to Robotics	MATH1025 3.00, MATH1310 3.00, EECS2031 3.00
EECS4422 3.00 Computer Vision	MATH1025 3.00, MATH1310 3.00, EECS2031 3.00
EECS4431 3.00 Advanced Topics in 3D Computer Graphics	EECS2021 4.00, EECS3431 3.00 (MATH1025 3.00 by transitivity from EECS3431 3.00)
EECS4441 3.00 Human Computer Interaction	EECS3461 3.00
EECS4452 3.00 Digital Signal Processing: Theory and Applications	EECS3451 4.00
EECS4461 3.00 Hypermedia and Multimedia Technologies	EECS3461 3.00
EECS4471 3.00 Introduction to Virtual Reality	MATH1025 3.00, MATH1310 3.00, EECS2021 4.00, EECS2031 3.00, EECS3431 3.00
EECS 4481 4.00 Computer Security Laboratory	EECS3221 3.00, EECS3214 3.00
EECS 4482 3.00 Computer Security Management: Assessment and Forensics	Any 12 university credits at the 3000 level in any discipline
EECS 4491 3.00 Simulation and Animation for computer Games	EECS3431 3.00, MATH1310 3.00 (MATH1025 3.00 by transitivity from EECS3431 3.00)

Project Courses

EECS4080 3.00 Computer Science Project	Permission of course coordinator; 36 EECS credits
EECS4081 6.00 Intelligent Systems Project	Open only to students in the Intelligent Systems Stream; permission of the instructor; EECS3401 3.00 and EECS3402 3.00 with gpa>= 6.00
EECS4082 6.00 Interactive Systems Project	Open only to students in the Interactive Systems Stream; permission of the instructor; EECS3311 3.00 and EECS3461 3.00
EECS4084 6.00 Communication Networks Project	Open only to students in the Communication Networks Stream; permission of the instructor; EECS3451 4.00 and EECS3213 3.00 with gpa>= 6.00
EECS4088 6.00 Computer Science Capstone Project	Permission of course coordinator; 36 EECS credits
EECS4090 6.00 Software Development Capstone Project	Only open to students in the Software Development Stream. EECS3311 3.00 with a B or better, EECS3101 3.00 and EECS3342 3.00
EECS 4480 3.00 Computer Security Project	Restricted to students in the Computer Security degree. Students must have passed 40 EECS credits. Permission of the course director is required
EECS 4700 6.00 Digital Media Project	Only open to students in the final year of the Digital Media program.

Degree Program Checklists

See URLs http://www.cse.yorku.ca/cscurrent_students/undergrad_students/index.html (current) or http://www.cse.yorku.ca/undergrad/csCalendars.html (archive)