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

## Preface

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

It is important, therefore, that you not only develop the practical and theoretical skills of a professional but that you also try to obtain an understanding of the impact of computers on society. For that reason we would strongly encourage you to select courses outside Computer Science and Engineering in areas where you will broaden your knowledge of society. 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.

So 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 which courses will make you a good professional. That implies a sound technical background, a broad education, professional ethics and a social conscience.

Lastly we would like to remind you that Computer Science or Computer Engineering is an art as well as a scientific/technical field and that means you cannot learn it entirely from a book — you must also practice it.

## The Department

Computer Science and Engineering Department
1003 Computer Science & Engineering Building (CSEB)
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)

| | |
|---|---|
| Undergraduate Director:<br>George Tourlakis | Tel. (416) 736-5334<br>Email: enquiries@cse.yorku.ca |
| Computer Engineering Director:<br>Robert Allison | Tel: (416)736-2100 X20192<br>Email: allison@cse.yorku.ca |
| Graduate Director: Rick Wildes | Tel. (416) 736-5053 |
| Chair: Amir Asif | Tel. (416) 736-5053<br>Fax: (416) 736-5872 |

## Faculty

| | Telephone Extension | email @cse.yorku.ca | | Telephone Extension | email @cse.yorku.ca |
|---|---|---|---|---|---|
| Aboelaze,Mokhtar | 40607 | aboelaze | Ma, Burton | 77885 | burton |
| Allison, Robert | 20192 | allison | Mackenzie, Scott | 40631 | mack |
| Amanatides, John | 44782 | amana | Mandelbaum, Marvin | 40630 | mandel |
| An, Aijun | 44298 | aan | Mirzaian, Andy | 70133 | andy |
| Arjomandi, Eshrat | 70130 | eshrat | Nguyen Uyen | 33274 | utn |
| Asif, Amir | 70128 | asif | Ostroff, Jonathan | 77882 | jonathan |
| Baljko, Melanie | 33348 | mb | Roosen-Runge, Peter (*Emeritus*) | | peter |
| Cercone, Nick | 55053 | nick | Roumani, Hamzeh | 66146 | roumani |
| Cribb, Peter | 70127 | peterc | Roventa, Eugene | 33928 | roventa |
| Datta, Suprakash | 77875 | datta | Ruppert, Eric | 33979 | ruppert |
| Dymond, Patrick | 33948 | dymond | Spetsakis, Minas | 77886 | minas |
| Eckford, Andrew | 70152 | aeckford | Stachniak, Zbigniew | 77877 | zbigniew |
| Edmonds, Jeff | 33295 | jeff | Stuerzlinger, Wolfgang | 33947 | wolfgang |
| Elder, James | 66475 | jelder | Toptsis, Anestis | 66675 | anestis |
| Godfrey, Parke | 66671 | godfrey | Tourlakis, George | 66674 | gt |
| Gotshalks, Gunnar | 33350 | gunnar | Tsotsos, John | 70135 | tsotsos |
| Gryz, Jarek | 70150 | jarek | Tzerpos, Vassilios | 33341 | bil |
| Hofbauer, John | 70125 | hofbauer | van Breugel, Franck | 77880 | franck |
| Hornsey, Richard | 33265 | hornsey | Vlajic, Natalija | 77878 | vlajic |
| Jenkin, Michael | 33162 | jenkin | Wallis, Anthony (*Emeritus*) | | wallis |
| Jiang, Hui | 33346 | hj | Wharton, Michael | 33978 | michael |
| Kant, Mariana | 70117 | mkant | Wildes, Richard | 40203 | wildes |
| Lesperance, Yves | 70146 | lesperan | Xu, Jia | 77879 | jxu |

## CSAC and CEAB Accreditation

The Computer Science Accreditation Council (CSAC) has accredited all Computer Science honours programs offered by the Department that have already graduated students, with the exception of the BA and BSc honours minor. The Computer Engineering specialized honours BASc program 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 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 recognised 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.

## 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. Computer Engineering
4. 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/calendars/index.htm). 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 The degree types are BA Honours, BSc Honours, BA or BSc Specialised Honours, International BSc (iBSc) and International BA (iBA).

The Honours major may be combined with most subjects in both LA&PS and Science and Engineering leading to a four-year double major or major-minor degree. Computer Science is also available as an Honours Minor program, which must be combined with an Honours Major in a different subject.

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 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 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 specialisation. However, a breadth in education is maintained by the requirement of a significant number of non-CSE and non-MATH courses.

A BA Honours or BSc Honours program requires 120-credits (normally completed in four years of study), more specialisation, a higher minimum performance (grade-point-

average of 5.0 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 Department also offers a 90-credit BA or BSc program normally completed in three years of study and requiring a minimum grade point average of 4.0 over all courses for graduation.

The recommended courses in computer science and mathematics are identical in most 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. Also all 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 degree by the end of the third year and can at that time graduate with either a BA or BSc. If you have the grade point average to be eligible for an honours program, you will be listed as an honours student for administrative purposes. Only the honours programs (with the exception of the minor) are accredited by the CSAC.

## Streams in Honours Computer Science Programs

Both the Specialised Honours and Honours Major programs may be taken with a specified focus or stream. The streams provide a mechanism for recognizing 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 choices within CSE courses that you would make yourself in an un-streamed Honours Major or Specialized Honours program), as well as a full year (6 credit) 4000–level project, or honors thesis as it would be called in some universities.

## 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 specialized 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.

---

[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 4.0 between 0-23 credits; 4.25 between 24-53 credits; 4.80 between 54-83 credits; 5.0 beyond 83 credits.*

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-CSE and non-MATH courses to ensure a breadth of general education.

## International Programs - iBSc and iBA Honours

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. This iBSc degree program requires 12 credits in a language stream chosen by the student and 18 credits that focus on a country or region that is compatible with the student's chosen language stream and/or international issue that is of interest to the student. The iBA requires 18 credits in a language stream chosen by the student and 12 credits that focus on a country or region that is compatible with the student's chosen language stream and/or international issue. 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 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, 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 requires a roughly equal number of courses from the Department of Computer Science and

Engineering and the Fine Arts Cultural Studies department of the Faculty of Fine Arts. There are also a few courses required from the Communication Studies program of the Division of Social Science in the Faculty of LA&PS.

For more information see the URL:
http://www.yorku.ca/web/futurestudents/programs/template.asp?id=652

### The Computer Engineering Program

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

Whereas 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.

Like all engineering programs the workload is very demanding. Total credits normally completed over four year amount to nearly 150, for example.

For more information see the URL:
http://www.eng.yorku.ca/current/programrequirements/index.html

## 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/calendars/index.htm

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 BASc.

The Department also provides degree checklists that itemize the course requirements in an accessible 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.

# Admission to Programs

## Computer Science and Computer Security Programs

Please go to http://www.yorku.ca/web/futurestudents/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://www.yorku.ca/web/futurestudents/requirements/highschool.html
Please note the Faculty-specific requirements as these pertain to your case.

2. Entry with post-secondary academic background

Please follow http://www.yorku.ca/web/futurestudents/requirements/univ_coll.html to find a detailed description of general University and Faculty-specific policies for gaining admission under this category.

In particular, 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 average of C+ or better if transferring to the honours computer science programs (minimum average of C is needed to transfer into the Bachelor degree programs)[2]

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://www.yorku.ca/web/futurestudents/requirements/.

## Computer Engineering Program

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

## Graduate Program in Computer Science

Admission to the graduate program is highly competitive. The ideal preparation for graduate studies in Computer Science is the completion of the Specialised Honours Program in Computer Science (please consult the Computer Science degree requirements, the degree checklist, and the course descriptions), or an equivalent degree (including 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

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

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

## Industrial Internship Program

The Technology Internship Program (TIP) offers qualified undergraduate Computer Science, Computer Security and Computer Engineering 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 and the length of an Internship can vary from four to sixteen months. During the work placement students earn a salary typical of entry-level positions in the IT sector.

Students in the BASc, BA Honours, BSc Honours, iBSc, iBA and Computer Security programs are eligible to apply. Students enrolled in the Internship option are required to enrol in CSE3900 0.00 (Internship Co-op Term) in *each term* of their internship.

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, 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 CSE3900 0.00 description 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.

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 announced content meshes with issues and problems studied in computer science.

Not only should you consider taking individual courses in other subjects but you should also consider taking a concentration of courses that together form a coherent or complementary package. Such a concentration may come from one discipline (one of the sciences, for example, because of their hierarchical structure) but it may also come from two or three disciplines on related concepts presented from different perspectives. It may also 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-CSE 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:

CSE1520 3.0 Computer Use: Fundamentals
CSE1530 3.0 Computer Use: Programming
CSE1540 3.0 Computer Use for the Natural Sciences
CSE1550 3.0 Computer Use: Web and Database Systems
CSE1560 3.0 Introduction to Computing for Math and Statistics
CSE1570 3.0 Introduction to Computing for Psychology

CSE1520 3.0 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 CSE1530 3.0 is an introduction to computer programming and may be taken as preparation for CSE1020 3.0 or for CSE2501 1.0, if the student lacks background in this area. CSE1550 3.0 offers a practical way of learning the basics of how information is specified, acquired, and managed using database technology.

Students taking the 1500 series courses are not eligible to take the 2000-level CSE courses for majors without successful completion of CSE1020 3.0 and CSE1030 3.0.

At the 2000-level the Department offers the course CSE2501 1.0, Fortran and Scientific Computing, which covers computer-based problem solving in a variety of scientific and engineering settings. As of 2005/06 we introduced two new service courses in C# programming, CSE2550 1.0 and CSE2560 1.0. These three 2500 series of courses have prerequisites.

## Recent Academic Changes

1. All the Programs offered by the Department will be housed in the Faculty of Science and Engineering as of 2009/10.
   In Fall 2009 all existing BSc and BA Computer Science and Computer Security programs will be housed in the Faculty of Science and Engineering (FSE).

2. New Programs and Program Changes (2009/10)
   Pending Senate approval, the Computer Science Minor has changed as described in the Degree Checklists section.

3. New Courses and Course Changes FW2009/10
   - New course: CSE1570 3.0 Introduction to Computing for Psychology (not offered in 2009/10)
   - New course: CSE2041 3.0 Net-Centric Computing (not offered in 2009/10)
   - CSE2031 3.0 Reformatted to two hours lectures and two hours weekly scheduled labs (Fall 09).

4. New Programs (2007/08)
   In Fall 2007 Senate approved the BA Specialised Honours Digital Media program and the iBA program. A brief description of these can be found above, in the section “Programs Offered by the Department”.

5. New Courses and Course Changes FW2008/09
   - CSE 1560 3.0 Introduction to Computing for Math and Statistics
   - CSE 1710 3.0 Programming for Digital Media
   - CSE 1720 3.0 Building Interactive Systems
   - CSE 3214 3.0 Computer Network Protocols and Applications
   - CSE 3403 3.0 Platform Computing
   - CSE 4213 3.0 Computer Networks II is no longer offered, replaced by CSE 3214 3.0
   - CSE 4425 3.0 Introductory Computational Bioinformatics
   - CSE 4214 4.0 Digital Communications - credit weighting changed from 3.0 to 4.0 with the addition of a scheduled lab and a tutorial
   - The following courses have scheduled labs (12 hours) added to them:
     CSE4210 3.0, CSE4215 3.0, CSE4352 3.0, CSE4421 3.0, CSE4422 3.0, CSE4431 3.0, CSE4452 3.0, CSE4471 3.0

6. New Program (2006/07)
   Senate approved a specialised honours degree program (both a BSc and BA version) in Computer Security. A brief description can be found above in the section Programs Offered by the Department.

## Student Clubs

The York University Computer Club (YUCC) is an organisation 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 organizes social events and advising sessions. They can be accessed though their website at http://engsoc.yorku.ca/.

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

The Women in Computer Science and Engineering (WiCSE) supports and promotes women in Computer Science and 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.

## The Student Ombuds Service

The Student Ombuds Service (SOS) is a peer-advising service designed to help York students — especially those in Bethune College and the Faculty of Science and Engineering — 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 214 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, by e-mail at sos@yorku.ca, or by phone at 416-736-5383.

## Computer Facilities

Undergraduate students who are registered to any CSE major course use the “Prism Laboratory” the Department of Computer Science and Engineering undergraduate computing laboratories. 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. Prism accounts can also be accessed remotely by dial-up, through the Internet via secure connection, or from other designated laboratories on campus including the Maxwell Lab, which is in the same building as our CSE laboratories. Selected labs are equipped with printing facilities. Senior students use a variety of specialty laboratories as required. These include the Robotics and Vision, the Digital Systems, the Software Engineering, 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, four Unix and one Windows 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. These displays are supported by a set of high-performance Linux workstations and the custom VE software environment developed at York.
- The Integrated Signal Processing and Multi-Media Laboratory consist of a number of Digital Signal Processing boards, each contain 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.

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.

## Computer Use Policy

Working in a laboratory situation 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.

# Computer Science and Computer Engineering Awards

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

### Allen S. Berg Award in Memory of Mark A. Levy

Up to five prizes will be awarded to outstanding Faculty of Science and Engineering students enrolled in third or fourth year computer science courses.

### Allen S. Berg Prize in Excellence in Engineering

Awarded to a student enrolled in Computer Engineering who has completed a minimum of 60 credits and has the highest cumulative grade point average.

### Nancy Waisbord Memorial Award

This is a cash award presented annually to a graduating student who has consistently demonstrated excellence in Computer Science.

### Computer Science Academic Achievement Award

Up to two cash awards will be presented to outstanding graduating students in an Honours program. These awards are funded by contributions from faculty members in the Department.

### Bursary in Science and Engineering

This bursary is available to assist Faculty of Science and Engineering students who are in financial need. Recipients must have completed at least 24 credits towards a BSc degree with a minimum cumulative grade point average of 5.0 on all science courses taken. Recipients must be Canadian citizens, permanent residents or protected persons, and Ontario residents, and demonstrate financial need.

### Business.Ca Inc. Bursary

Students in their third year, enrolled in the Faculty of Science and Engineering, in good academic standing are eligible for this award. Applicants must be Canadian citizens, permanent residents or protected persons, Ontario residents and demonstrate financial need.

### Prestigious Awards

The Faculty of Science and Engineering also awards various medals to its top-graduating students. These include the Gold Medal of Academic Excellence (Faculty of Science and Engineering).

### Other Awards

- Students in the Department are encouraged to apply for Summer Science 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 7.0(B+) is required. In addition, faculty sometimes employ undergraduate research

assistants over the summer period. While not an award administered by NSERC, such positions are only offered to the best students in the Department.

- Hany Salama Bursary – a cross-Faculty bursary available to ITEC, MATH and CSE students, administered by Student Financial Services.
- Sally Murray Findley Memorial Scholarship – a cross-Faculty scholarship available to ITEC, MATH and CSE students, administered by Student Financial Services.

# Academic Policies

## Advising

Academic advising is available on an individual or a group basis in the Department. Group advising provides help in choosing courses to fulfil degree requirements. Individual faculty advising is available to discuss relevant academic issues such as recommended mathematical skills, theoretical versus applications oriented courses, areas of specialisation, graduate studies and career paths.

It is ultimately the responsibility of each student to ensure that they meet all degree requirements of both the Department and their home Faculty (i.e., Science and Engineering). Written information and program check lists are provided to assist you in making appropriate choices. It is recommended that you take advantage of advising opportunities to answer any questions you may have.

Group advising is scheduled by year level during March and early April. In addition, individual advising appointments may be made through the Undergraduate Office.

## Academic Honesty

The University Senate, the Faculty of Science and 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 academically dishonest. The deliberate presentation of part of another student's program text or other work as your own without acknowledgment is academically dishonest, and renders you liable to the disciplinary procedures instituted by the Faculty of Science and Engineering.

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/calendars/index.htm

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 is a relatively young and rapidly changing discipline. 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 computer science is the year in which you take your first major CSE course, and it normally coincides with the year you were admitted into the program. If you have a break in your studies then your starting year changes to the year in which you start taking major CSE courses again. Since most Senate approved degree program regulations become effective in the fall term following 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 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.

- 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. 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, 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 standard Departmental form, obtained from the Undergraduate Office, *within 21 days of receiving notification of the grade.*
For more details on the University's reappraisal policies see
http://www.registrar.yorku.ca/services/policies/grade.htm).

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 than the original grade. The student will receive the report of the petitions committee 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 Executive Committee of the Faculty.

## Grading System

Grading at York University is done on a letter scale. 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 and techniques in satisfying the requirements of a piece of work or course.
- A (8) Excellent — Thorough knowledge of concepts and/or techniques together with a high degree of skill and/or some elements of originality in satisfying the requirements of a piece of work or course.
- B+ (7) Very Good — Thorough knowledge of concepts and/or techniques together with a fairly high degree of skill in the use of those concepts and techniques in satisfying the requirements of a piece of work or course.
- B (6) Good — Good level of knowledge of concepts and/or techniques together with a considerable skill in using them in satisfying the requirements of a piece of work 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 a piece of work 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 a piece of work 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 a piece of work or course.
- D (2) Barely Passing — Minimum knowledge of concepts and/or techniques needed to satisfy the requirements of a piece of work 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 you need to have before taking the course.

Prerequisites are enforced and if you enroll in a course for which you do not meet the prerequisite you will be removed as early as possible after the start of the course. Depending on office workload this may be up to the end of the sixth week of the term.

Prerequisites include both specific courses and at the 3000- and 4000-level a GPA over CSE courses of 4.5 or better. “General Prerequisites” is a term used to describe prerequisites that apply to (almost) every course at a particular level, and is used simply to avoid having to repeatedly specify the same thing! Thus, for example, for 3000- level courses CSE2011 3.0 and the CSE GPA of 4.5 or better are general prerequisites.

## Fees

All courses have an associated fee of $10.00, with the following exceptions: All 4000-level courses; all service courses; CSE1019 3.0, CSE3001 1.0, CSE3002 1.0, CSE3121 3.0, CSE3122 3.0 and CSE3900 0.0. This fee is to offset consumable costs associated with operating the PRISM lab. This includes paper, toner, and maintaining and servicing printers within the lab.

The cost of these fees will be reviewed from year to year and adjusted accordingly. The associated course fee will not normally be refunded, but will be refunded if you withdraw from the course before the first lecture or because of Departmentally initiated de-enrolment.

## Course Weights

Courses normally meet for three class hours a week for one term (these are 3 credit–courses whose numbers end in "3.0"). Some courses have required one-hour labs per week (e.g., CSE1020 3.0 and CSE1030 3.0). Catalogue numbers are assigned to the labs rather than the lectures and students use the REM to enrol by selecting an appropriate lab. Other courses have a similar registration system and lab requirements, but the associated labs are three hours per week, which entails a 4.0 weight for the course (e.g., CSE2021, 3201, 3215 and 3451 are “4.0 courses”). Some of the 3.0 courses at the 2000 and 3000 levels have optional tutorials. All CSE courses put heavy demands on the student’s time by requiring the completion of take-home assignments or projects.

## Service Courses

Courses with second digit 5 (e.g. 1520, 1530, 1540, 1550, 1560, 2501, 2550, 2560) may be taken to satisfy Faculty degree requirements but do not count as major

credits, and the grades from such courses are not included in calculating the prerequisite grade point average.

In what follows, if a course is referenced by its *rubric* (e.g., MATH, CSE) and number only, then the Faculty prefix "SC/" is implied. If the prefix is different than this, then it will be explicitly given, e.g., "AK/COSC3503 3.0", "AP/ITEC1620 3.0".

## Course Descriptions: 1000-Level

### CSE 1019 3.0 Discrete Mathematics for Computer Science (Cross listed with MATH 1019 3.0)

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-$\Omega$, big-$\Theta$ 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 3.0, or both of 12U Advanced Functions & Introductory Calculus and 12U Geometry & Discrete Mathematics.
*Course Credit Exclusion*: MATH2320 3.0

CSE 1020 3.0 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, scaleable, and maintainable software. Based on this, CSE1020 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 lecture hours 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 CSE1520, CSE1530 or CSE1540 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 CSE1520 3.0 and CSE1530 3.0 instead (see the following descriptions).

*Prerequisites*: One of (1) – (4) below must be met:

(1) (New high school curriculum): Two 12U Math courses including advanced functions and introductory calculus with minimum mathematics average of 75% on the two courses, and no mathematics grade below 65%.

(2) (Old high school curriculum): *OAC calculus* and one other OAC in mathematics (normally ***finite mathematics*** or ***algebra & geometry***) with an average grade of 75% in all OAC mathematics and no grade less than 65%.

(3) Completion of 6.0 credits from York University MATH courses (not including AK/MATH1710 6.0 or courses with second digit 5) with a grade average of 5.0 (C+) or better over these credits;

(4) Completion of AK/MATH1710 6.0, or 6.0 credits from York University mathematics courses whose second digit is 5, with an average grade not below 7.0 (B+).

Strongly Recommended: Previous programming experience; for example, a high school programming course or CSE1530 3.0.

*Course Credit Exclusion*: AP/ITEC1620 3.0

### CSE 1030 3.0 Introduction to Computer Science II

This course continues the separation of concern theme introduced in CSE1020. While CSE1020 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.

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

*Prerequisites:* CSE1020 3.0
*Course Credit Exclusion*: AP/ITEC2620 3.0

### CSE 1520 3.0 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 organisation (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 CSE1020 *3.*0/1030 3.0: *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 CSE1020 3.0.
*Note*: This course counts as elective credits towards satisfying Faculty degree requirements but does not count as Computer Science major credits.

### CSE 1530 3.0 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 CSE1020 3.0 and CSE1030 3.0. This course does not count as a Computer Science major credit.

*Prerequisites:* None
*Course Credit Exclusion*: CSE1540 3.0
*NCR Note:* No credit will be retained if this course is taken after the successful completion of or simultaneously with CSE1020 3.0 or AS/AK/ITEC1620 3.0

### CSE 1540 3.0 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 Engineering and students in the BA Applied Math program.

Note: This course is not open to any student who has passed or is taking CSE1020 3.0. 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 Exclusion:* CSE1530 3.0
*NCR Note:* No credit will be retained if this course is taken after the successful completion of or simultaneously with CSE1020 3.0 or CSE2501 1.0

CSE 1550 3.0 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 organisations. 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 Exclusion*: SB/OMIS 3730 3.0
*NCR Note:* No credit will be retained if this course is taken after the successful completion of, or simultaneously with CSE3421 3.0 or ITEC3220 3.0.

*Note*. This course does not count for Computer Science major credit.

### CSE 1560 3.0 Introduction to 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 trigonometry functions and examples.

*Prerequisites:* AS/SC/AK/MATH 1300 3.0

*Co requisites*: AS/SC/AK/MATH 1310 3.0; AS/SC/AK/MATH 1131 3.0

*NCR Note:* No credit will be retained if this course is taken after the successful completion of, or simultaneously with SC/PHYS 2030 3.0

### CSE 1570 3.0 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 6.0
*Course Credit Exclusions:* CSE1530 3.0, CSE1560 3.0

### CSE 1710 3.0 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 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:* CSE1530 3.0, ITEC1620 3.0
*NCR Note:* No credit will be retained if this course is taken after the successful completion of, or simultaneously with CSE1020 3.0

### CSE 1720 3.0 Building Interactive Systems

This course continues an introduction to computer programming within the context of image, sound and interaction, subsequent to CSE1720 3.0. 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:* CSE1710 3.0
*Course Credit Exclusions:* CSE1020 3.0, ITEC1620 3.0, ITEC1630 3.0

## Course Descriptions: 2000-Level

### General Prerequisites

- CSE1030 3.0 with a grade of C+ or better

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

### CSE 2001 3.0 Introduction to the Theory of Computation

The course introduces different theoretical models of computers. Topics covered may 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, CSE1019 3.0

### CSE 2011 3.0 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, CSE1019 3.0

### CSE 2021 4.0 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

### CSE 2031 3.0 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 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

## CSE 2041 3.0 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 3.0

### CSE 2501 1.0 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; optimisation; differential equations; Fourier transform
- Simulation: random numbers; distributions; queues
- Monte Carlo method
- Processing experimental data
- Data visualisation
- Chaos and fractals

*Prerequisites*: CSE1020 3.0 or CSE1530 3.0

### CSE 2550 1.0 Introduction to C# Programming

Introduction to the C# programming language: programming constructs analogous to those taught in CSE1030 3.0; basic data structures if time permits.

- Comparison of C# vs. Java and C, C++. The VisualStudio.NET development environment. C# program structure.
- Control structures (if/else, while, for, switch, break, continue)
- Operator overloading
- Arrays, strings
- Exception handling
- Classes, methods, namespaces, parameter passing, method and constructor overloading, inheritance, polymorphism, interfaces, abstract classes.

*Prerequisites*: CSE1030 3.0 or ITEC2620 3.0
*NCR note:* This course is not open for credit to students who passed CSE3403 3.00

*Note:* Does not count for major credit for computer science, or towards engineering requirements.

### CSE 2560 1.0 C# Programming Tools for Graphical User Interfaces

Introduction to programming graphical user interfaces (GUI) in the C# programming language: building GUIs in C# under the VisualStudio.NET IDE; the major GUI components and event handling mechanism of C#.

- GUI development: General, C# and .NET specific
- Events
- Building Windows applications: Forms
- GUI components of C#: Labels, TextBoxes, Buttons, GroupBoxes, Panels, CheckBoxes, RadioButtons, PictureBoxes, Menus, LinkLabels, ListBoxes. More advanced features as time permits (for example, colour control, font control, drawing, imaging, animation).

*Prerequisite: CSE2550 1.0*
*NCR note:* This course is not open for credit to students who passed CSE3403 3.0
*Note:* Does not count for major credit for computer science, or towards engineering requirements.

## Course Descriptions: 3000-Level

### General Prerequisites

- CSE2011 3.0
- A cumulative grade point average of 4.5 or better over all completed[3] major computer science courses and CSE1019 3.0

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.

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 CSE courses may be taken in any one of the fall or winter terms 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 CSE2031 3.0 before entering third year.*

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

### CSE 3000 3.0 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 1.0, PHYS 3001 1.0, CSE3001 1.0

### CSE 3101 3.0 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
- Amortisation: 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.
- Jeff Edmonds' notes: "How to Think about Algorithms". Available Online <http://www.cse.yorku.ca/%7Ejeff/notes/3101/notes.html>

*Prerequisites:* General prerequisites, CSE2001 3.0, MATH1090 3.0, MATH1310 3.0

CSE 3121 3.0 Introduction to Numerical Computations I
(Cross-listed with AS/SC/MATH 3241 3.0)

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 factorisation 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 factorisation 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:* CSE1540 3.0 or CSE2031 3.0 or SC/CSE2501 1.0; MATH1010 3.0 or MATH1014 3.0 or MATH1310 3.0; MATH1025 3.0 or MATH1021 3.0 or MATH2021 3.0 or MATH2221 3.0

CSE 3122 3.0 Introduction to Numerical Computations II
(Cross-listed with AS/SC/MATH3242 3.0)

The course is a continuation of CSE3121 3.0. 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:* CSE3121 3.0

### CSE 3201 4.0 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:* General prerequisites, CSE2021 4.0. PHYS3150 3.0 is strongly recommended

### CSE 3213 3.0 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 3.0
*Course Credit Exclusion:* COSC3211 3.0

### CSE 3214 3.0 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 Exclusion:* CSE4213 3.0

### CSE 3215 4.0 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
- Input/Output ports, Interrupts, and timers
- Memory systems
- Analog to digital and digital to analog conversion
- Parallel and Serial Interfacing
- Hardware Modelling
- Structural specification of hardware
- Synthesis of combinational circuits using a Hardware Description Language
- Synthesis of sequential circuits using a Hardware Description Language
- Rapid Prototyping using field programmable gate arrays

*References:*


- Michael D. Ciletti, Modelling, Synthesis, and Rapid Prototyping with the VERILOG (TM) HDL, 1/e, Prentice-Hall, ISBN 0-13-977398-3 1.
- Richard E. Haskell, Design of Embedded Systems Using 68HC12/II Microcontrollers, Prentice-Hall, ISBN 0-13-083208-1.
- Frank Vahid and Tony Givargis, Embedded System Design: A Unified Hardware/Software Introduction, John Wiley & Sons, ISBN: 0471386782.

- John B Peatman, Design with Microcontrollers, Prentice Hall, ISBN 0-13-759259-0
- The 8051 Microcontroller 3/e. Prentice-Hall, ISBN 0-13-780008-8.

*Prerequisites*: General prerequisites; CSE2031 3.0; CSE3201 4.0

### CSE 3221 3.0 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, CSE2021 4.0, CSE2031 3.0
*Course Credit Exclusion:* COSC3321 3.0

### CSE 3301 3.0 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, CSE2001 3.0

### CSE 3311 3.0 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, CSE2001 3.0, CSE2031 3.0, MATH1090 3.0

### CSE 3341 3.0 Introduction to Program Verification

Every program implicitly asserts a theorem to the effect that if certain input conditions are met then the program will do what its specifications or documentation says it will. Making that theorem true is not merely a matter of luck or patient debugging; making a correct program can be greatly aided by a logical analysis of what it is supposed to do, and for small pieces of code a proof that the code works can be produced hand-in-hand with the construction of the code itself. Good programming style works in part because it makes the verification process easier and this in turn makes it easier to develop more complex algorithms from simple ones.

The course will provide an introduction to the basic concepts of formal verification methods. It will also include the use of simple tools to aid in verification.

Topics covered will include the following:

- The role of formal verification in the software life cycle; verification vs. testing and validation

- Introduction to propositional calculus; checking for tautologies and contradictions; annotating code with assertions
- Symbolic execution; proving relative correctness for small code segments; establishing termination
- Creating specifications with quantifiers; translating specifications into code

*Suggested reading*:

- Gries and Schneider, A Logical Approach to Discrete Mathematics, Springer-Verlag, 1993.
- R. Backhouse, Program Construction and Verification, Prentice-Hall, 1986

*Prerequisites:* General prerequisites, MATH1090 3.0
*Course Credit Exclusion:* COSC3111 3.0

### CSE 3401 3.0 Functional and Logic Programming

This course covers functional and logic programming. Together with the students' background on procedural and object-oriented programming, the course allows them to compare the development of programs in these different types of languages.

"Functional programs work with values, not states. Their tools are expressions, not commands. How can assignments, arrays and loops be dispensed with? Does not the outside world have states? These questions pose real challenges. The functional programmer can exploit a wide range of techniques to solve problems." (Paulson, 1996)

"Based on predicate logic, it [logic programming] allows computing problems to be expressed in a completely `declarative' way, without giving instructions for how the problem is to be solved. An execution mechanism, like the one embodied in implementations of Prolog, can then be used to search efficiently and systematically for a solution of the problem." (Spivey, 1996)

Topics on functional programming may include: recursive, polymorphic and higher-order functions; recursive types and type inference. Topics on logic programming may include backtracking, resolution and unification.

*Prerequisites:* General prerequisites, MATH1090 3.0

### CSE 3402 3.0 Introduction to Concepts of Artificial Intelligence

Artificial Intelligence (AI) deals with building a system that can operate in an intelligent fashion. Neat as this simple definition is, it obscures the complex nature of intelligence. At the time of the Dartmouth Conference (1956), regarded by many as the start of AI, some researchers believed it would be possible to create a "thinking machine" in a matter of a few years. That was close to 40 years ago, and we are still far from our goal, but we have learned a lot on the way.

In this course, we begin by discussing differing definitions of artificial intelligence and go on to examine fundamental concepts in AI, building on material introduced in CSE3401 3.0: Functional and Logic Programming. Topics to be covered include reasoning under uncertainty, search, constraint propagation, planning and problem solving.

*Prerequisites:* General prerequisites, CSE3401 3.0

### CSE 3403 3.0 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

### CSE 3421 3.0 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 3.0, ITEC3220 3.0, ITEC3421 3.0

### CSE 3431 3.0 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, visualisation, or real-time rendering.

*Prerequisites*: General prerequisites, CSE2031 3.0, MATH1025 3.0
*Course Credit Exclusion*: GL/CSLA3635 3.0

### CSE 3451 4.0 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.

In this course, students review the concept of a discrete signal, the conditions under which a continuous signal is completely represented by its discrete version, linear time-invariant systems.

Topics covered may include the following:

- Continuous and discrete signals
- Linear time-invariant systems
- Fourier analysis in continuous time
- Fourier analysis in discrete time
- Sampling
- Filtering, image enhancement
- Laplace transform
- Z transform
- Linear feedback systems
- Random signals, image coding
- Kalman filtering
- Statistical pattern recognition

There are three supervised lab hours per week.

*Prerequisites:* General prerequisites, MATH1310 3.00
*Course Credit Exclusions:* COSC4242 3.0, COSC4451 3.0, EATS4020 3.0, MATH4130B 3.0, MATH4830 3.0, PHYS4060 3.0

### CSE 3461 3.0 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 Exclusion:* AP/ITEC3230 3.0, AP/ITEC3461 3.0
*NCR Note:* No credit will be retained by students who successfully completed AS/SC/COSC4341 3.0 or AS/SC/COSC4361 3.0 before FW99.

### CSE 3481 3.0 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, CSE3213 3.0

### CSE 3900 0.0 Internship Co-op Term

The objective of the course is to provide qualified students a hands-on, practical work experience that formally integrates the student's academic knowledge with real-world situations in a "co-operative" work setting. Enrolment in the course is mandatory in each term that a student undertakes a work placement. Students will be assigned a faculty supervisor, although the Internship Co-op Coordinator and the Internship Office will take the lead in placement and interaction with the placement site.

*Prerequisites*:
Successful completion of at least 9.0 computer science credits at the 3000 level including CSE3311 3.0 (Software Design) and an overall GPA of at least 6.0 in

Mathematics and Computer Science courses completed[4]. To qualify, ***in the first instance***, the student must be enrolled full-time in the Honours Program and attend all mandatory preparatory sessions as outlined by the Internship Co-op Office.

*Notes:*

- This course does not count for degree credit in any program. Registration in sections of CSE3900 while on an internship placement provides a transcript notation of the student's participation in the internship program.
- Students are required to register in this course in every term of their work term (internship co-op).
- Every student registered in the course will be assigned a faculty supervisor who will assess the student's performance during the internship.
- Successful applicants will have 18 credits remaining to complete their honours degree upon enrolment to the program.

*Evaluation:*

Performance in each term (CSE3900 0.0) 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 Co-op 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.

---

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

General Prerequisites

- CSE2011 3.0
- A cumulative grade point average of 4.5 or better over all completed[5] major computer science courses and CSE1019 3.0

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.

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

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

### CSE 4080 3.0 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*: CSE4001 6.0, CSE4081 6.0, CSE4082 6.0, CSE4084 6.0

### CSE 4081 6.0 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

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

organisational 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 CSE3401 3.0 and CSE3402 3.0 with a minimum grade of B, and have prior permission of the instructor.

*Course Credit Exclusions*: CSE4001 6.0; CSE4080 3.0; CSE4082 6.0; CSE4084 6.0

### CSE 4082 6.0 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 organisational 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 CSE3311 3.0 and CSE3461 3.0, and have prior permission of the instructor.

*Course Credit Exclusions*: CSE4001 6.0, CSE4080 3.0, CSE4081 6.0, CSE4084 6.0

### CSE 4084 6.0 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 organisation 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 CSE3451 4.0 and CSE3213 3.0, and have prior permission of the instructor.
*Course Credit Exclusions*: CSE4001 6.0, CSE4080 3.0, CSE4081 6.0, CSE4082 6.0

### CSE 4090 6.0 Software Development 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 CSE3311 3.0, and completion of CSE3101 3.0, CSE3221 3.0, CSE3401 3.0, and CSE3341 3.0
*Co requisites*: CSE 4312 3.0, CSE 4313 3.0
*Course Credit Exclusions*: none

CSE 4101 3.0 Advanced Data Structures (integrated with CSE5101 3.0)

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:

*Prerequisites:* General prerequisites; CSE3101 3.0.

CSE 4111 3.0 Automata and Computability (integrated with CSE5111 3.0)

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; CSE3101 3.0

CSE 4115 3.0 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 diagonalisation, complexity hierarchies, resource bounded reducibility such as log space and polynomial time reducibility
- P vs. NP: Nondeterminism, Cook's Theorem and techniques for proving NP-Completeness
- Nondeterministic space: The Savitch and Immerman/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; CSE3101 3.0

CSE 4161 3.0 Mathematics of Cryptography
(Cross-listed with MATH 4161 3.0)

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 CSE3101 3.0, or permission of the instructor.

CSE 4201 3.0 Computer Architecture (integrated with CSE5501 3.0)

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; CSE3201 4.0, CSE3221 3.0

### CSE 4210 3.0 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; CSE3201 4.00; CSE3451 3.00

### CSE 4211 3.0 Performance Evaluation of Computer Systems
(integrated with CSE5422 3.0)

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 3.0, CSE3213 3.0

### CSE 4214 4.0 Digital Communications

Digital communications has become a key enabling technology in the realisation 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, synchronisation, 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 Quantisation
- 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)
- Synchronisation 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; CSE3213 3.0; MATH2030 3.0; one of CSE3451 4.0, EATS 4020 3.0, MATH 4830 3.0, PHYS 4060 3.0, or PHYS 4250 3.0
*Course Credit Exclusion:* CSE4214 3.0

### CSE 4215 3.0 Mobile Communications

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, localisation, 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; CSE3213 3.0

CSE 4221 3.0 Operating System Design (integrated with CSE5421 3.0)

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; CSE3221 3.0
*Course Credit Exclusion:* COSC4321 3.0

CSE 4301 3.0 Programming Language Design (integrated with CSE5423 3.0)

This course is a continuation of CSE3301 3.0 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; CSE3301 3.0

CSE 4302 3.0 Compilers and Interpreters (integrated with CSE5424 3.0)

Principles and design techniques for compilers and interpreters. Compiler organisation, compiler writing tools, scanning, parsing, semantic analysis, run-time storage organisation, memory management, code generation, and optimisation. 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
- Optimisation: local and global optimisations; dead code removal; control flow analysis

*Prerequisites:* General prerequisites; CSE3301 3.0 recommended

### CSE 4311 3.0 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 CSE3311 3.0 or CSE3221 3.0

### CSE 4312 3.0 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 nonfunctional requirements
- Pragmatic requirements engineering: Technology transfer; Traceability
- Current Requirements Standards, e.g., IEE 830 Recommended Practice for Requirements Engineering
- Requirements Categorisation 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; CSE3311 3.0

### CSE 4313 3.0 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; CSE3311 3.0

CSE 4351 3.0 Real-Time Systems Theory (integrated with CSE5441 3.0)

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; CSE3221 3.0

CSE 4352 3.0 Real-Time Systems Practice (integrated with CSE5442 3.0)

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; CSE3221 3.0

CSE 4401 3.0 Artificial Intelligence (integrated with CSE5326 3.0)

This course will be an in-depth treatment of one or more specific topics within the field of Artificial Intelligence. Possible topics include the following:

- Machine learning: deduction, induction, and abduction, explanation-based learning, learning k-DNF
- Statistical learning: reinforcement learning, genetic learning algorithms, and connectionist learning systems, supervised and unsupervised
- Statistical and structural pattern recognition
- Speech recognition
- Artificial intelligence programming paradigms: search, pattern-directed inference, logic- and object-oriented programming, symbolic mathematics, constraint satisfaction and symbolic relaxation, building problem solvers, efficiency issues
- Sensor-based robotics: path planning, position estimation, map building, object recognition, robotic sensor and actuator hardware, software, and interfacing

Contact the course director for information regarding the focus of the course this year.

*Prerequisites:* General prerequisites; CSE3402 3.0

CSE 4402 3.0 Logic Programming (integrated with CSE5311 3.0)

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; CSE3401 3.0; one of CSE3101 3.0 or CSE3341 3.0

### CSE 4411 3.0 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; CSE2021, CSE2031, CSE3421 3.0

### CSE 4412 3.0 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; CSE3101 3.0; CSE3421 3.0; one of MATH2030 3.0 or MATH1131 3.0

CSE 4413 3.0 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; CSE3213 3.0; CSE3421 3.0

CSE 4421 3.0 Introduction to Robotics (integrated with CSE5324 3.0)

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 3.0, MATH1310 3.0, CSE2031 3.0

CSE 4422 3.0 Computer Vision (integrated with CSE5323 3.0)

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 3.0; MATH1310 3.0; CSE2031 3.0

CSE 4425 3.0 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

CSE 4431 3.0 Advanced Topics in 3D Computer Graphics (integrated with CSE5331 3.0)

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; CSE2021 4.0; CSE3431 3.0; MATH1310 3.0
*Course Credit Exclusion:* COSC4331 3.0

CSE 4441 3.0 Human Computer Interaction (integrated with CSE5351 3.0)

- 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 Visualisation (Queries, Visualisation, 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; CSE3461 3.0
*Course Credit Exclusion:* COSC4341 3.0

### CSE 4452 3.0 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; CSE3451 3.0

### CSE 4461 3.0 Hypermedia and Multimedia Technology

The course focuses this year on the design and implementation of hypermedia presentation systems. "Hypermedia" refers to the non-linear organisation 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; CSE3461 3.0
*Course Credit Exclusion*: COSC4361 3.0

### CSE 4471 3.0 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, visualisation & 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 3.0; MATH1310 3.0; CSE2021 4.0; CSE2031 3.0; CSE3431 3.0 (may be waived on an individual basis, please consult the instructor).
Course credit exclusion: COSC4471 3.0

### CSE 4481 4.0 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 authorisation; 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; CSE3221 3.0, CSE3481 3.0

### CSE 4482 3.0 Computer Security Management: Assessment and Forensics (not offered until 2009/10)

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

CSE 4700 6.0 Digital Media Project (not offered until 2010/11)

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 Computer Science and Engineering 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*: CSE4080 3.0; CSE4081 6.0; CSE4082 6.0; CSE4084 6.0

## Degree Program Checklists

These are found on the page

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

under the heading "Current Dept. degree checklists".