Faculty	Telephone Extension	Email @cs.yorku.ca
M. Aboelaze	40607	aboelaze
R. Allison	20192	allison
J. Amanatides	44782	amana
E. Arjomandi	70130	eshrat
P. Cribb	70127	peterc
S. Datta	77875	datta
P. Dymond	33948	dymond
J. Edmonds	44298	jeff
P. Godfrey	70128	godfrey
G. Gotshalks	77875	gunnar
J. Gryz	70150	jarek
M. Jenkin	33977	jenkin
Y. Lesperance	70146	lesperan
J. Liu	33928	joseph
M. Mandelbaum	40630	mandel
J. Mason	66671	jmason
A. Mirzaian	70133	andy
J. Ostroff	77882	jonathan
R. Paige	77878	paige
P. H. Roosen-Runge	77844	peter
H. Roumani	66146	roumani
E. Ruppert	33979	ruppert
H. Sandhu (on leave)		hsandhu
M. Spetsakis	77886	minas
Z. Stachniak	77877	zbigniew
W. Stuerzlinger	33947	wolfgang
A. Toptsis	66675	anestis
G. Tourlakis	66674	gt
J. Tsotsos	70135	tsotsos
F. van Breugel	77880	franck
A. Wallis	77874	tony
R. M. Wharton	33978	michael
J. Xu	77879	jxu

Table of Contents **Page**

Preface

In choosing to study Computer Science you have chosen a career in an exciting and rapidly changing discipline. As a computer scientist, 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 computer scientist 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 your elective courses outside Computer Science 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 an area outside of Computer Science.

So in planning your course selection you should be thinking ahead and asking yourself not only which courses will give you a good Computer Science degree, but which courses will make you a good professional computer scientist. That implies a sound technical background, a broad education, professional ethics and a social conscience. You can't get all that in your first year but you can at least make a start.

Lastly we would like to remind you that Computer Science is an art as well as a science which means you cannot learn it entirely from a book - you must also practice it. **We recommend a maximum of three Computer Science courses per term.**

The Department

Computer Science Department
York University
4700 Keele Street
Toronto, Ontario M3J 1P3
www.cs.yorku.ca

Office hours 9:00-12:00, 2:00-4:00

Gunnar Gotshalks, Undergraduate Director		Tel. (416)736-5334
	Email:	enquiries@cs.yorku.ca
Minas Spetsakis, Graduate Program Director		Tel. (416)736-5334
		www.cs.yorku.ca/grad/
Michael Jenkin, Chair		Tel. (416)736-5053
		Fax: (416)736-5872

CSAC Accreditation

All Computer Science honours programs offered in the Faculty of Pure and Applied Science and the Faculty of Arts, with the exception of the BA and BSc honours minor, are accredited by the Computer Science Accreditation Council (CSAC). Accreditation for honours programs at Atkinson Faculty of Liberal and Professional Studies is pending.

The Computer Science Accreditation Council is an autonomous body established by the Canadian Information Processing Society (CIPS). The purpose of accreditation is to identify those institutions that offer computer programs worthy of recognition. The objectives of the Council 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 ISP designation is formally recognized by the provinces of Ontario and Alberta. More information on professional accreditation and the accreditation process can be found on the CIPS web page at www.cips.ca.

A Note on Terminology

In this document BA or BSc degree refers to the 90-credit bachelor degree. BA(Hons) or BSc(Hons) refers to the 120-credit degree.

Admission to the Computer Science Major

The educational background of students who seek admission to a Computer Science degree program generally belongs to one of two categories. The requirements for each are outlined below.

In each case a student must first have been admitted to an undergraduate program in either the Faculty of Arts, Atkinson Faculty of Liberal and Professional Studies or the Faculty of Pure and Applied Science. Each Faculty has certain admission requirements that must be met. The requirements described below are in addition to these faculty admission requirements (which are not described here).

1. **Entry with only secondary school background**
   If a student's only academic background is at the OAC level admission to a Computer Science major requires:
   - an overall OAC average of 77%.
   - at least 2 mathematics OACs (one must be calculus) with an average over of 75% with no mathematics grade less than 65%.

   For students from outside Ontario with only secondary school credits the York University Admissions Office will assess the equivalency of that educational background.

2. **Entry with post-secondary academic background**
   This category includes the following:
   - students who are already admitted to York University and wish to change their major.
   - students who transfer from other universities in Canada.
   - students who have completed courses at any post-secondary educational institution anywhere in the world.
   - students who have completed courses at a community college in Canada or CEGEP in Quebec.

   Admission to a Computer Science degree program requires:
   - a B (6.0) average over all courses taken.
   - at least 6 credits of mathematics, with a B (6.0) average over all major stream mathematics credits taken and/or a B+ (7.0) average over all service stream mathematics credits.

   The York University Admissions Office will assess the conversion of grades from other institutions to the York University grade scale.

If you have any post-secondary education you may not gain admission under category 1 above.

Changing your Major to Computer Science?
Please note that category 2 above applies to you if you are already a York University student and want to change your major - that is:
- a B (6.0) average over all courses taken at York University.
- at least 6 credits of mathematics, with a B (6.0) average over all mathematics credits taken and/or a B+ (7.0) average over all service stream mathematics credits.

NOTE: Students who have a post-secondary education from an institution other than York University, and who then take courses at York University before applying to major in Computer Science, must meet the B (6.0) average in York University courses. That is, the B average will be applied to both the previous academic record and to the York University academic record.

Access to Courses

York Enrolment System

Students enrol in courses using the York Enrolment System, via either a telephone or Web interface, typically in the few months prior to the start of each term. Computer Science courses frequently reach their class size maximum, in which case the following procedures are followed.

See www.cs.yorku.ca/undergrad/enrollment-guidelines.html for an expanded description and interpretation of the enrollment policy outlined below.

Application for Normal progress

We are committed to ensuring that students can make timely progress towards meeting degree requirements. However, we cannot normally accommodate students taking more COSC courses than they need, and neither can we normally accommodate students wishing to repeat a course they either dropped or failed, or wishing to take additional courses to make up for ones they dropped or failed.

Normal progress is defined as follows:

- Taking 1000-level courses in calendar year one, 2000-level in calendar year two, etc. There is no guarantee that students will be able to take 3000-level computer science courses in the same term as they are completing a 2000-level computer science course for example.
- Taking two 2000-level courses per term; or three 3000- and 4000-level courses per term
- When close to graduation being able to take necessary courses within the limits specified above.

Limits on Course Enrolment

A maximum of two 2000-level Computer Science courses in one term, and three 3000- or 4000-level courses, is permitted. In the summer term students are not permitted to take more than 6 credits in Computer Science.

Removal from Courses

If any student enrolls in more than the allowed number of courses per term they will be removed from whichever courses the department requires space. The Department also reserves the right to move students from a course in one term to the same course in the next term should such steps be necessary to ensure equitable access to courses. This includes movement from fall to winter or winter to summer.

Prerequisites
Students are responsible for ensuring they enrol only in courses for which they meet the prerequisites. Prerequisites include a minimum GPA over computer science courses. Students will be removed from a course if they do not meet the prerequisites, at any time before or during the course.

Courses Outside the Department
Students wishing to take Computer Science courses at another institution should submit a Letter of Permission (LOP) form. For the purpose of satisfying departmental degree requirements, the number of Computer Science (COSC) credits taken outside the Department of Computer Science may not exceed 6 credits in core courses and 12 credits in total. Advanced standing credit is included as credit taken outside the Department.

Definition of Core Courses:
Core courses include all 1000- and 2000-level Computer Science major courses, COSC3101 3.0, COSC3221 3.0, COSC3311 3.0, MATH1300 3.0, MATH1310 3.0, MATH1090 3.0, MATH2090 3.0.

Recent & Current (00/01) Academic Changes

1. **New and Recent Changes to Degree Requirements**
   - Beginning FW01/02 Bachelor degree programs will require MATH2090 3.0 (instead of MATH2320 3.0), and COSC3101 3.0, COSC3221 3.0 and COSC3311 3.0 as partial fulfillment of the 3000-level COSC breadth requirement.
   - Beginning FW00/01 all degree programs required COSC2031 3.0 in addition to those 2000-level COSC courses previously required. For honours degrees the required 3000-level COSC credit total was reduced by 3.
   - Beginning FW00/01 all honours degree programs required the following specific courses at the 3000-level to satisfy breadth – COSC3101 3.0, COSC3221 3.0, COSC3311 3.0, and one Applications area (Group A) COSC34xx 3.0 course. For 90-credit degrees the 3000-level COSC breadth requirement was unchanged.

2. **Reclassification of 3000-Level Courses**
   Group A and Group B classification of 3000-level courses in each group is removed. Consequently the COSC34xx 3.0 course required for breadth in the Applications area can be any COSC34xx course.

3. **New Courses or Course Numbering Changes**
   COSC3213 3.0, Computer Networks I
   COSC4213 3.0, Computer Networks II
   COSC4221 3.0, Operating Systems (formerly COSC4321 3.0)
   COSC4431 3.0, Computer Graphics (formerly COSC4331 3.0)
   COSC4441 3.0, Human Computer Interaction (formerly COSC4341 3.0)

COSC4451 3.0, Signals and Systems (formerly COSC4242 3.0)
COSC4461 3.0, Hypermedia and Multimedia Technology (formerly COSC4361 3.0)

4. **Deleted Courses**
COSC3211 3.0 Data Communications
COSC3212 3.0 Computer Networks

Programs Offered

For detailed information you are advised to first read the appropriate sections of the York University Undergraduate Calendar (click on Calendars in the York University web page - www.yorku.ca); secondly, read this supplemental Calendar, and thirdly, see an advisor in the Department of Computer Science at one of the regularly scheduled advising sessions.

Computer Science is available as a major program leading to an Honours (120-credit) degree in either the Faculty of Arts, the Faculty of Pure and Applied Science or Atkinson Faculty of Liberal and Professional Studies. It may also be combined with most subjects in both Arts and Science leading to a four-year double major or major-minor degree. These degree types are BA(Hons) or BSc(Hons)

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.

Bachelor (90-credit) vs. Honours (120-credit) Programs
A BA or BSc program requires 90-credits (normally completed in three years of study) and a grade point average of 4.0 over all courses. A BA(Hons) or BSc(Hons) program requires 120-credits (normally completed in four years of study), more specialization, a higher minimum performance (a grade-point-average of 5.0), and in some cases different courses than an BA or BSc degree.

All programs are structured in such a way that a student who embarks on a BA(Hons) or BSc(Hons) 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 (5.0), 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.

Specialized Honours
Students selecting this program take more courses in computer science and mathematics than for other programs in Computer Science thereby achieving greater specialization. However, a breadth in general education is maintained by the requirement of a significant number of non-COSC and non-MATH courses.

Space and Communication Sciences Stream
This is a specialized honours BSc stream in computer science combined with a concentration of courses in the Departments of Earth and Atmospheric Science, and Physics and Astronomy. Students select courses on knowledge-based programming, numerical methods, data communications, electronics, space communications and physics of the space environment. Fourth year features electives from an extensive list of topics from all three departments.

Entry is highly competitive, as the first year is limited to approximately 40 places. Candidates are required to have an A average in high school. It is also a very demanding program, as students must maintain a Science grade point average of 6.0.

BSc or BA Honours Double Major or Honours Major/Minor
The intention of a combined program is for students to major in two subjects while maintaining a 5.0 average. 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, but still generally includes courses at the 4000-level. Such degrees may 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.

BA Inter-Faculty Honours Double Major
It is also possible to combine a computer science major as a student in the Faculty of Arts with a major in the Faculty of Pure and Applied Science. This leads to a BA(Hons) degree. Not all combinations of two majors are permitted.

BA Honours Double Major Program in Computer Science and Mass Communications Studies
This double major program differs from a standard double major program in that the second major is in an interdisciplinary program. In this double major program, students are required to complete at least 42 credits in computer science courses, 6 credits of which must be at the 4000 level. Students are also required to complete 36 credits in Mass Communications Studies, 6 of which must be at the 4000 level.

Elective Courses

Students in Computer Science 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 other courses, even though about a quarter of their courses are electives. 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. 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 emphasize the importance of elective courses, all honours programs, except the BA and BSc minor, now require at least 30 credits from non-COSC and non-MATH courses.

Industrial Internship Program

The internship program offers qualified undergraduate and graduate Computer Science students the opportunity to take part in a program that alternates academic studies with related work experience in either the private or public sectors. An Internship normally lasts sixteen months but students spend a minimum of four months at an employers work location. There is considerable flexibility in the duration of individual Internships and the length of an Internship can vary from 4 to 16 months. For a sixteen-month placement, students at York University on average earn about $45,000.

The Internship program begins in the third year of study for undergraduates and first year for graduates. Students in either the BA(Hons), BSc(Hons), MSc or PhD programs can enroll. For undergraduates a minimum average of B is required and students must be full time students at York University in order to be considered for the Internship program. Students enrolled in the Internship option normally go on the Internship between their 3rd and 4th years. Interested students should inquire about the program after their second year of study.

The department maintains an Internship Office to assist students seeking internship employment and to assist employers wishing to hire York University Internship students. The Internship office coordinates recruitment activity on campus. Internship students receive assistance in identifying relevant and interesting internship opportunities, formulating the employer application package and sharpening the interview skills. Students are placed at a wide range of companies including IBM Canada Ltd., Nortel Networks, and Microforum.

BA Information Technology Program

The Faculty of Arts Information Technology Programme (ITEC) is designed to provide students with the ability to examine how information and computer technology interact with culture and society. The multi-disciplinary core of the program combines the applied aspects of computer systems with the historical, social, and ethical contexts of computing and information processing and dissemination.

Drawn from the various components of the program, the skills the ITEC program fosters include computing, problem solving, analytical, research, and critical writing skills. The program offerings are structured around both technology related courses - which develop the applied aspects of computer systems - and non-technology related courses - which focus on understanding the implications of technology across a broad range of activities within our society.

Broadly speaking ITEC focuses on the application of computer systems and their impact on society and culture whereas computer science focuses more on the creation of those computer systems.

Because the ITEC program requires the introductory computer science (major) courses it is feasible for students to change their major either from COSC to ITEC or from ITEC to COSC. In either case there is some deficit (courses not taken) that would have to be made up during the first term or so in the new program.

Admission to the Graduate Program in Computer Science

Admission to the MSc program is highly competitive.

The ideal preparation for graduate studies in Computer Science is the completion of the Specialized Honours Program in Computer Science in the Faculty of Pure and Applied Science at York University (please consult the Computer Science degree requirements, the degree checklist, and the course descriptions), or its equivalent (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 admission. Of course, the higher your grades the more likely you will be a successful candidate.

To request an application kit send e-mail to gradinfo@yorku.ca requesting an application package for computer science. Include a physical (street, city etc) mail address in your e-mail message.

Need to upgrade a degree?
If you already have a Computer Science degree then, if necessary, you would upgrade your background to be equivalent to the Specialized Honours Program in Computer Science. A comparison of the degree program you completed with the

Specialized Honours program will show you what you are missing. If you have a 90-credit degree, then you will need to upgrade your degree to the Honours level.

It is recommended that you become familiar with the Unix, C/C++ and the X-window system environment.

How to upgrade a degree

Obtain the Undergraduate Program Supplemental Calendar and the Graduate Program FAQ sheet from CCB125 (or the web site). Compare the courses you have taken in your previous degree(s) with the descriptions of Computer Science courses at York University, checking off on the Specialized Honours degree checklist form those courses that you think are very similar to ones you have already taken as part of your previous degree(s).

Count how many Computer Science courses you would need to take for the Specialized Honours degree. If this number is greater than 4 go to the York Admissions Office (Atkinson College building, room 150) and apply for admission to the undergraduate degree program. If the number is 4 or less go to the York Admissions Office and apply for admission as a special undergraduate student.

There is no need to make an appointment with the Undergraduate Program Director or the Graduate Program Director. Neither person can officially tell you how many courses in the undergraduate program you will get credit for, and they cannot estimate it any better than you can yourself.

The Service Program

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

The starting courses for non-majors are COSC1520 3.0, COSC1530 3.0, Introduction to Computer Use I & II and COSC1540 3.0, Computer Use for the Natural Sciences. The course COSC1530 3.0, Introduction to Computer Use II is an introduction to computer programming and may be taken as preparation for COSC1020 3.0 if the student lacks background in this area. Students taking the 1500 series courses are not eligible to take the 2000-level Computer Science courses without successful completion of COSC1020 3.0 and COSC1030 3.0.

York University Computer Club

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

Computer Facilities

Undergraduate students use the Ariel Lab, the Department of Computer Science undergraduate computing laboratories. The lab servers can be accessed remotely by dial-up and through the Internet. In the lab itself first and second year students have access to 37 colour NCD X-terminals, and 23 Sun Ultra 10 workstations. Third and fourth year students are granted access to the Senior Lab consisting of 20 Sun Ultra 10 workstations. Senior students may also use a variety of specialty laboratories in their courses including the Robotics Laboratory, the Real-Time Laboratory, and the Multimedia Laboratory.

- The Robotics Laboratory consists of two CRS robot arms, a wireless iRobot Magellan Pro autonomous mobile robot and 6 Sun Ultra 10 workstations equipped with multimedia hardware including video and audio facilities.
- The Digital Logic Laboratory provides hands-on experience in computer design.
- The Real-Time Laboratory provides a high-performance Sun Ultra 60 workstation, an Intel-based PC and industry-standard software tools for the design and analysis of real time systems. The laboratory has a Marklin digital train set with computer controlled and monitored locomotives, turnouts and position sensors. The Sun workstation and the PC are used to control the set.
- The Multi-media Laboratory supports 3D graphics, audio input/output, virtual reality hardware, a large screen projector, and a 6 degree-of-freedom tracker. The lab consists of five SGI NT-based workstations and five Macintosh G4 workstations.

All workstations and computers in the Department are connected up to the campus network backbone, providing access to all significant systems in the University, as well as computers around the world via Internet.

Access to the Ariel Lab machines requires an authorized account and a password, as issued by the Department. Each student receives an Ariel account, providing a personal space for storing files, electronic mail, WEB publishing, course work, and access to printing facilities.

Students can also access their accounts remotely from other designated labs on campus (some with 24-h access) or from home computers.

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 computer science 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 authorized 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 (see the link Official York Policies, under Administrative Services at www.yorku.ca)

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 computer science 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 offenses. 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 of Computer Science does not assume any liability for damages caused by such activities.

Computer Science Awards

Unless otherwise stipulated students in both the Faculty of Pure and Applied Science and the Faculty of Arts are eligible for these awards. Plaques commemorating the achievement awards are maintained by the department.

Mark A. Levy Computer Science Award
Up to five prizes will be awarded to outstanding Faculty of Pure and Applied Science students enrolled in third or fourth year computer science courses.

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.

Other Awards
Students in the Department are encouraged to apply for Summer Science awards. 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 average of 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.

Prestigious Awards
The Faculties of Arts and Pure and Applied Science also award various medals to their top graduating students. These include the Governor General's Silver Medal (Faculty of Arts) and the Gold Medal of Academic Excellence (Faculty of Pure and Applied Science).

Atkinson College awards
Students whose home Faculty is Atkinson College are also eligible for the following scholarships and bursaries:
- Computer Science Major Program Scholarships
- Atkinson College Students' Association Scholarship
- Hany Salama Bursary
- Sally Murray Findley Memorial Scholarship

See the Atkinson College Calendar for details.

Academic Policies

Advising

Academic advising is available on an individual or a group basis in the Department of Computer Science. Group advising provides help in choosing courses to fulfil degree requirements. Individual faculty advising is available to discuss academic issues relevant to computer science such as recommended mathematical skills, theoretical versus applications oriented courses, areas of specialization, 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 the Faculty of Pure and Applied Science, or the Faculty of Arts, or Atkinson College. 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 Faculty of Arts, Faculty of Pure and Applied Science, Atkinson College, 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 piece of work is submitted by a student it is expected that all unquoted and uncited ideas (except for common knowledge) and text are original to the student. Uncited 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 Pure and Applied Science.

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 own 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 and Senate policies 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 nonacademic 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
Whenever new program requirements are introduced the following policies apply:

- The starting year in computer science is defined as the first academic year in which you took or will take COSC1020 3.0, if you take courses in consecutive years. If you have a break in your studies then your starting year changes to the year in which you are readmitted.
- If requirements change you may continue with your studies using the 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 from your starting year.
- If requirements change you may elect to graduate under the new requirements but you must meet all of them. You are not permitted to mix and match old and new requirements.

Appeal Procedures

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

With respect to a formal appeal, there are different procedures for course work and for final examinations and final grades. Of necessity, a formal appeal must involve only written work.

Course Work
An appeal against a grade assigned to an item of course work must be made ***within 14 days of the grade being made available.***

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.***

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 which 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.

Course Descriptions : 1000-Level

Courses in Computer Science have three class hours a week for one term (3 credit–course numbers end in "3.0"), unless otherwise indicated. Courses with second digit 5 (e.g. 1520, 1530, 1540, 3530) may be taken to satisfy Faculty degree requirements but do not count as Computer Science major credit, and the grades from such courses are not included in calculating the Computer Science prerequisite grade point average.

COSC 1020 3.0
Introduction to Computer Science I (same as AS/AK/ITEC1020 3.0)
Introduction to computation, computing machinery, algorithms and programming via theoretical concepts and practical skills. Problem solving via the structure, design and analysis of algorithms and their implementation as effective, correct and efficient programs. Control and data structures of a structured programming language (Java).

This course is introductory to the discipline in that it is the first in a hierarchy of courses; it is not a survey course. The emphasis is on the development of a theoretical conceptual basis and the acquisition of the intellectual and practical skills required for further study. The course is intended for prospective computer science 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 whose interest is casual, nor for those who require remedial work in the necessary background.

Warning: The work for this course includes a substantial number of exercises which require problem analysis, program preparation, testing, analysis of results, 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. During the first few weeks there is a scheduled laboratory. After that students book time in the computer laboratory on an as needed basis.

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 COSC1520 3.0 and COSC1530 3.0 instead (see the following descriptions).

Prerequisites: If no university-level mathematics: OAC Calculus and one other OAC in mathematics (normally Finite Mathematics or Algebra and Geometry) with an average grade of 75 percent in all OAC mathematics and no grade less than 65 percent; otherwise: at least 6 credits of university-level mathematics with a grade average over all MATH credits of C+ or better [B+ or better if it is a service course (second digit is 5) or AK/MATH1710 6.0].
Strongly Recommended: Previous programming experience; for example, a high school programming course or COSC1530 3.0.
Degree Credit Exclusion: AK/COSC2410 6.0, AK/COSC2411 3.0

COSC 1030 3.0
Introduction to Computer Science II (same as AS/AK/ITEC1030 3.0)
This course is a continuation of COSC1020 and covers some of the fundamentals of software development, various data structures (arrays, queues, stacks, trees, lists), and algorithms that use these structures (sorting, searching). An object oriented approach will be introduced. Students will use the Unix operating system with the X Window System.

Prerequisites: COSC1020 3.0
Degree Credit Exclusion: AK/COSC2410 6.0, AK/COSC2412 3.0

COSC 1520 3.0
Introduction to Computer Use I
This course is appropriate for students who are **not majoring in Computer Science**, but who would like an introduction to the use of the computer as a problem-solving tool. No previous computing experience is assumed, but the course does involve extensive practical work with computers, so some facility with problem-solving and symbolic operations will be very helpful.

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

Topics to be studied include: the development of information technology and its current trends; analysis of problems for solution by computers, report generation, file processing; spreadsheets; 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 *COSC1020 3.0/1030 3.0: Introduction to Computer Science.*

Note: This course is not open to students who have passed or are taking COSC1020 3.0. This course counts as elective credits towards satisfying Faculty degree requirements but does not count as Computer Science major credits.

Prerequisites: none

COSC 1530 3.0
Introduction to Computer Use II
Concepts of computer systems and technology - e.g. software engineering, algorithms, programming languages, theory of computation. 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 majors. However, it may be used as preparation by those who wish to major in Computer Science but lack programming background. Students who plan to major in Computer Science must also take COSC1020 3.0 and COSC1030 3.0. This course does not count as a Computer Science major credit.

Prerequisites: none
Degree Credit Exclusions: COSC1540 3.0. This course is not open to any student who has passed or is taking COSC1020 3.0.

COSC 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 Pure and Applied Science and students in the BA Applied Math program.

Note: This course is not open to any student who has passed or is taking COSC1020 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.
Degree Credit Exclusion: COSC1530 3.0

Course Descriptions: 2000-Level

General Prerequisites
Before enrolment is permitted in any 2000-level computer science course the following must be met.

- COSC1030 3.0 completed.
- MATH1090 3.0 completed
- A cumulative grade point average of 4.5 or better over completed Computer Science courses (including only the most recent grades in repeated courses).

Specific prerequisites may also apply to individual courses. Taking more than two 2000-level Computer Science courses per term is not permitted.

COSC 2001 3.0
Introduction to Theory of Computation
The course introduces different theoretical models of computers. Topics covered may include the following.

- Finite automata and regular expressions. Practical applications ie. text editors.
- Pushdown automata and context-free grammars. Practical applications e.g. parsing and compilers.
- Turing machines. Turing machines as a general model of computers. Introduction to the halting problem and NP completeness.

Prerequisites: general prerequisites.

COSC 2011 3.0
Fundamentals of Data Structures (same as AS/AK/ITEC2011 3.0)
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.

COSC 2021 3.0
Computer Organization (same as AS/AK/ITEC2021 3.0)
Computers can be usefully viewed as having a structure organized into several levels, ranging from high-level programming languages such as Java to digital logic circuits. Each level provides specific resources and abstractions for the programmer, which are created by the level beneath it.

This course provides students with basic understanding of computers at the low-lying levels of this structure. This includes programming in assembly / machine language, computer organization (CPU, DRAM, I/O, and busses), CPU structure (Datapath and Control), and Digital Logic. The presentation is centered on performance and covers topics like caching, pipelining, and parallel processing. The course presents theoretical concepts as well as concrete implementations on a modern, RISC processor.

Required Textbook:
- Patterson, D. and Hennessy, J., Computer Organization and Design: The Hardware / Software Interface, 2nd Edition, Morgan Kaufmann Publishers (1997).

Suggested Reading:
- Tanenbaum, A.S., *Structured Computer Organization,* 5th ed., Prentice-Hall, 1999.
- Stallings, Wm., *Computer Organization and Architecture,* 5th ed., Macmillan, 2000.

Prerequisites: general prerequisites.

COSC 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.

Suggested Readings:
- Kernighan and Ritchie, The C Programming Language (ANSI C Edition).

- Kernighan and Pike, The Practice of Programming

Prerequisites: general prerequisites.

Course Descriptions: 3000-Level

General Prerequisites

- COSC2011 3.0 completed.
- One of COSC2001 3.0 or COSC2021 3.0 or COSC2031 3.0 completed.
- A cumulative grade point average of 4.5 or better over completed Computer Science courses (including only the most recent grades in repeated courses).
- MATH1300 3.0 and MATH1310 3.0 completed.
- One of MATH2090 3.0, MATH1025 3.0, or MATH2320 3.0 completed.

Specific prerequisites may also apply to individual courses.

Warning: **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 COSC2031 3.0 before entering third year.**

COSC 3001 1.0
Organization and Management Seminar in Space and Communication Sciences (same as SC/EATS3001 1.0 and SC/PHYS3001 1.0)
A seminar course taught by guest speakers from industry, government and the university. Content changes from year to year, but includes such topics as professional ethics, communications regulations, space law, space science policy, project management, privacy and security issues in computing.

Prerequisites: Eligibility to proceed in the Specialized Honours stream in SCS beyond the 2000-level requirements.
Degree Credit Exclusions: EATS 3001 1.0, PHYS 3001 1.0, COSC3002 1.0

COSC3002 1.0
Organization and Management Seminar
A seminar course taught by guest speakers from industry, government and the university. Content changes from year to year, but includes topics such as professional ethics, communications regulations, project management, privacy and security, legal issues in computing.

Prerequisites: general 3000-level prerequisites
Degree Credit Exclusions: EATS 3001 1.0, PHYS 3001 1.0, COSC3001 1.0

COSC 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 are 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, randomized quicksort and its average case analysis, decision tree lower bounds, linear-time selection.
- Divide-and-conquer: binary search, quicksort, mergesort, polynomial multiplication, arithmetic with large numbers.
- Dynamic Programming: matrix chain product, scheduling, knapsack problems, longest common subsequence, some graph algorithms.
- Greedy methods: activity selection, some graph algorithms.
- Amortization: the accounting method, eg, 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*, McGraw-Hill and The MIT Press, 1991.
- P. Gloor, S. Dynes, I. Lee, *Animated Algorithms CD-ROM*, The MIT Press 1993.
- D.E. Knuth, *The Stanford GraphBase: A platform for combinatorial computing,* Addison-Wesley & The ACM Press, 1993.

Prerequisites: general prerequisites, including MATH2090 3.0 or MATH2320 3.0 or MATH2442 3.0
Degree Credit Exclusion: AK/COSC3432 3.0

COSC 3121 3.0
Introduction to Numerical Computations I
(same as 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 factorization for symmetric systems are explained. Orthogonal transformations are studied for computations of the QR decomposition and the Singular Values Decompositions (SVD). The use of these transformations in solving linear least squares problems that arise from fitting linear

mathematical models to observed data is emphasized. Finally, polynomial interpolation by Newton's divided differences and spline interpolation are discussed as special cases of linear equations. The emphasis of the course is on the development of numerical algorithms, the use of intelligent mathematical software and the interpretation of the results obtained on some assigned problems.

Topics covered may include the following:

- Preliminaries - linear algebra, computer programming and mathematical software
- Number systems and errors - machine representation of numbers, floating-point arithmetic, simple error analysis, ill-conditioned problems and unstable algorithms
- Solution of systems of linear equations - Gaussian elimination and its computational complexity, pivoting and stability, special structures (Cholesky's factorization for positive definite systems, banded systems, storage and computational complexities) error analysis, condition number and iterative refinement
- Solution of overdetermined 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: COSC1540 3.0 or COSC2031 3.0 or AK/COSC3501 3.0; 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.
Degree Credit Exclusions: AK/COSC3511 3.0, MATH3241 3.0

COSC 3122 3.0
Introduction to Numerical Computations II
(same as AS/SC/MATH3242 3.0)
The course includes a study of algorithms and computer methods for differentiation, integration, and solution of ordinary differential equations. Nonlinear equations of one variable, systems of nonlinear equations, optimization of functions of one and several variables and their relation to nonlinear equations are also covered. 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:
- Solution of nonlinear equations and unconstrained optimization - single nonlinear equation; systems of nonlinear equations; unconstrained optimization.

- Numerical differentiation and integration - methods of estimating derivatives; error analysis for differentiation; the rectangle and trapezoid rule for integration; Simpson's rule; Romberg's integration; adaptive quadrature routines; truncation and round-off errors in integration; improper integrals.
- Solution of ordinary differential equations - introduction; analytical versus numerical solutions; basic numerical methods; Euler's, Heun's methods; Taylor series methods; order of a method; local and global errors; Runge-Kutta methods; Predictor-corrector methods; systems of differential equations; boundary value problems.

Prerequisites: COSC3121 3.0; MATH2270 3.0
Degree Credit Exclusion: MATH3242 3.0

COSC 3201 3.0
Digital Logic Design

- Boolean algebra and logic gates. How complex functions on data at the bit-representation level can be built up from simple primitives such as And, Or and Not (or just Nand or Nor). Implementation of logic functions using these primitives. Families of logic circuits.
- Combinatorial circuits, implementing functions whose output depends only on their inputs. Canonical forms of Boolean functions and their simplification using Karnaugh maps and the Quine-McClusky method. Complex combinatorial units, such multiplexers, encoders, arithmetic-logic units(ALU), read-only memory(ROM), and programmable arrays.
- Sequential circuits, implementing functions whose output depends on their history as well as their current input. Construction of basic, clocked, master-slave, and edge-triggered flip-flops. Higher level (register-transfer) constructs such as registers, counters and read-write memory (RAM).
- Theoretical design concepts, such as finite state machines.
- Hands-on digital logic hardware laboratory.

Suggested readings:

- John Hays, *Introduction to Logic Design*, Addison Wesley, 1993.
- M.M. Mano, *Digital Design*, Prentice Hall, 1991.

Prerequisites: general prerequisites, including COSC2021 3.0.

COSC 3213 3.0
Computer Networks I

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
Degree Credit Exclusions: COSC3211 3.0, COSC3212 3.0, AK/COSC3409A 3.0, AK/COSC3409B 3.0, ITEC3210 3.0

COSC 3221 3.0
Operating System Fundamentals
(formerly COSC3321 3.0 - before S2000)
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, including COSC2021 3.0; COSC 2031 3.0
Degree Credit Exclusion: COSC3321 3.0, AK/COSC3470 3.0

COSC 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 emphasized. Examples are drawn from early and contemporary programming languages, including Fortan, 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, 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, exceptions.
- Object-oriented language features: classes, encapsulation, inheritance, polymorphism.
- Critical and comparative evaluation of programming languages.

Prerequisites: general prerequisites, including COSC 2001 3.0.

COSC 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, 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 modeling designs; alternative modeling languages like UML, data-flow diagrams, structure charts, etc.
- static software modeling; dynamic modeling and behavioural modeling.
- 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, including COSC 2001 3.0 and MATH2090 3.0 and COSC2031 3.0

COSC 3331 3.0
Object-Oriented Programming and Design
Introduction to the theoretical and practical methods of object-oriented software construction. Topics include single and multiple inheritance, type hierarchies, polymorphism, operator overloading, object persistence, class library design, generic classes, and design by contract.

This course is a detailed introduction to the methodology of object-oriented software construction, one of the major areas of software engineering practice and research.

An object-oriented program is based on classes. Instances of classes, called objects, are created as the program runs and interact through message passing. Messages that are acceptable to an object are defined by the interface of the class. Use of interfaces promotes the reuse of reliable code and can make it easier to carry out maintenance and extension.

The course will introduce the theoretical concepts of object-oriented programming and design, and present examples using the object-oriented programming language Eiffel. Eiffel will be compared to other OO languages, like C++, Java, Smalltalk, Python, and Delphi. The course will cover more OO techniques in far more detail than in previous courses, and the students will apply these techniques in a large group design and programming project.

Topics to be covered in the course will include the following:
- Abstract data types as a basis for information hiding and as a theory for OO software construction.
- Other software engineering principles such as the open-closed principle and the single choice principle.
- Information hiding; secret vs. public vs. selectively available information.
- Single inheritance: subtyping; substitutability; subcontracting.
- Feature adaptation: changing export status; renaming methods; ambiguity in inheritance; overriding; redefinition of behaviour; breaking the subtype relationship.
- Design by contract: preconditions, postconditions, class invariants, and their use in designing software systems. Debugging using contracts. Testing using contracts.
- Documenting OO programs; automatic documentation tools.
- Exception handling.
- Generic (parameterized) classes as a mechanism for code reuse.
- Polymorphism, dynamic binding, and static binding.
- Type checking: static vs. dynamic checking.
- Principles for inheritance: inheritance vs. use.
- Multiple inheritance for class library design. Repeated inheritance.

- Persistence. Linking OO programs with a database.
- Class interfaces: partial implementations vs. full implementations.

Prerequisites: general prerequisites
Degree Credit Exclusion: COSC3010A 3.0; AK/COSC3650 3.0

COSC 3341 3.0
Introduction to Program Verification
(formerly COSC3111 3.0 - before S2000)

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

Prerequisites: general prerequisites, including MATH 2090.03
Degree Credit Exclusion: COSC3111 3.0

COSC 3401 3.0
Introduction to Symbolic Computation

The course will introduce and explore programming concepts used in symbolic and knowledge-based computing. It is intended to give the student a programming background which will be useful for further work in logic programming, expert systems, and artificial intelligence.

The programming language Prolog will be considered in detail. Prolog is a declarative programming language based on the concept of a logical assertion. It is widely used for constructing knowledge-based and expert systems.

The course will develop the following concepts.
- Terms as representations of facts
- Logical clauses as rules
- Recursive programming techniques
- Backward-chaining vs. forward-chaining
- Goal search through backtracking
- Building logical databases for knowledge-based problem-solving
- Representing mathematical knowledge by rewrite rules
- Natural language processing using grammar rules

Prerequisites: general prerequisites, including MATH 2090 3.0

COSC 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 COSC3401 3.0: Introduction to Symbolic Computation. Topics to be covered include reasoning under uncertainty, search, constraint propagation, planning and problem solving.

Prerequisites: general prerequisites; COSC3401 3.0; MATH2090 3.0 or MATH2320 3.0 or MATH2442 3.0

COSC 3408 3.0
Simulation of Discrete Systems
Simulation is a technique for dealing with problems that do not admit exact (or "analytic") solutions via mathematical analysis. A model of the system to be studied is constructed, and then the model is run to see how it performs, either to predict how the system will behave, or, if the behaviour of the system is known, to test the validity of the model of the system. A computer is a tool for supporting a large amount of activity in the running of the model.

A "discrete system" simulation is one which admits a discrete-event model that can be run in discrete steps that match the structure of the model. (For simulation of continuous systems see COSC 3418 3.0)

Examples of discrete systems studied by simulation include games and other dynamic systems involving small populations where it is feasible to model individual's behaviour. Major sub-topics include the generation and use of random numbers, queuing systems, and the visual presentation of behaviour.

Prerequisites: general prerequisites; MATH2560 3.0.
Degree Credit Exclusion: MATH4930B 3.0

COSC 3418 3.0
Simulation of Continuous Systems
Simulation is a technique for dealing with problems that do not admit exact (or "analytic") solutions via mathematical analysis. A model of the system to be studied is constructed, and then the model is run to see how it performs, either to predict how the system will behave, or, if the behaviour of the system is known, to test the validity of the model of the system. A computer is a tool for supporting a large amount of activity in the running of the model.

A "continuous system" may either be presumed to be inherently continuous or it may, at a fine enough scale, be actually composed of discrete events. However, in simulation, a "continuous system" is one for which the model, due to practical necessity, is described by continuous variables regardless of its physical structure. However, the running of a continuous model involves, also of necessity, discrete steps. Thus central to continuous system simulation is the problem of approximation. (For simulation of discrete systems see COSC 3408 3.0)

Examples of continuous systems studied by simulation include dynamic systems involving very fine variations or large populations. Major sub-topics include chaotic behaviour, the numerical solution of differential equations by finite methods, and related issues of stability and errors.

Prerequisites: general prerequisites; MATH2560 3.0.

COSC 3421 3.0 (same as AS/AK/ITEC3421 3.0)
Introduction to Database Systems
Concepts, approaches and techniques in database management systems (DBMS). Logical model of relational databases. An introduction to relational database design. Other topics such as 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

Prerequisite: ITEC/COSC2011 3.0
Degree Credit Exclusion: AS/SC/COSC3412 3.0

COSC3461 3.0 (same as AS/AK/ITEC3461 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 utilize modern toolkits and scripting languages to implement UIs. One of the assignments focuses on user interface evaluation.

Prerequisites: ITEC/COSC2011 3.0 or COSC2031 3.0
Degree Credit Exclusion: Not open to students who successfully completed AS/SC/COSC4341 3.0 or AS/SC/COSC4361 3.0 before FW99.

Course Descriptions: 4000-Level

General Prerequisites
Before enrolment is permitted in any 4000-level computer science course the following requirements must be met.

- COSC2001 3.0, COSC2011 3.0, COSC2021 3.0, COSC2031 3.0 completed.
- at least 12 credits in COSC courses at the 3000-level completed.
- a cumulative grade point average of 4.5 or better over completed computer science courses (including only the most recent grades in repeated courses).
- MATH2090 3.0 completed

Specific prerequisites may apply to individual courses.

COSC 4001 6.0 (same as SC/EATS4001 6.0 and SC/PHYS4001 6.0)
Space and Communication Sciences Workshop
Individual projects will be assigned by mutual agreement between the student and a faculty member. The work may be done under supervision by the faculty member or under supervision of an industrial associate to that faculty member. The projects will be self-contained problems of a design nature, and will be pursued in the manner of a space project. Thus, the first step is to define the requirements of the design, the second to carry out a review of previous work, and the third to execute the design. Following that, the design shall be tested, normally through simulation, and conclusions drawn. A report of professional quality shall be written and submitted.

Prerequisites: Satisfactory completion of the 3000-level courses in the Space and Communication Science core
Degree Credit Exclusions: COSC4080 3.0, EATS4001 6.0, PHYS4001 6.0

COSC 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 completed six full computer science courses. 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.

Internship students may elect to receive credit for their internship as a project course. This is outlined further at the beginning of this calendar. A 'contract' is still required.

Prerequisites: general prerequisites; permission of the course director. Restricted to students who have completed 36 credits in Computer Science.
Degree Credit Exclusion: COSC4001 6.0

COSC 4101 3.0 (x-listed COSC5101 3.0)
Advanced Data Structures
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 amortized analysis. If

time permits, some lower bound techniques may be discussed, as well as NP-completeness proof techniques and approximation algorithms.
The course may include the following topics:
- Amortized and worst-case analysis of data structures.
- Data structuring paradigms: self-adjustment and persistence.
- Lists: self-adjustment with the move-to-front heuristic.
- Search trees: splay trees, finger search trees.
- Heaps: skew heaps, Fibonacci heaps.
- Union-find trees.
- Link-and-cut trees.
- Multidimensional data structures and dynamization.

Prerequisites: general prerequisites, including COSC3101 3.0 or COSC3432 3.0

COSC 4111 3.0 (x-listed COSC 5111 3.0)
Automata and Computability
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, including COSC3101 3.0 or COSC3432 3.0

COSC 4201 3.0 (x-listed COSC 5501 3.0)
Computer Architecture
This course presents the core concepts of computer architecture and design ideas embodied in many machines and emphasizes 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, including COSC 3201 3.0, and COSC3221 3.0 (or COSC3321 3.0)

COSC 4211 3.0 (x-listed COSC5422 3.0)
Performance Evaluation of Computer Systems
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 modeling 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, including COSC 3211 3.0 or COSC3213 3.0; and COSC3408 3.0

COSC 4213 3.0
Computer Networks II
More advanced topics in networking, concentrating on higher-level protocols, security, network programming and applications. Topics covered may include the following:

- Higher level protocols
- Security: encryption, authentication, firewalls
- Network programming; RPC
- Multimedia: compression and multimedia standards
- The web: URL/http, CGI programming, dynamic html, proxy servers, wireless networks

Prerequisites: general prerequisites, including COSC3212 3.0 or COSC3213 3.0

COSC 4221 3.0 (x-listed COSC 5421 3.0)
Operating System Design
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, including COSC 3221 3.0 or COSC3321 3.0.
Degree Credit Exclusion: COSC4321 3.0

COSC 4301 3.0 (x-listed COSC5423 3.0)
Programming Language Design
This course is a continuation of COSC3301 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 nonalgorithmic 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, including COSC 3301 3.0

COSC 4302 3.0 (x-listed COSC5424 3.0)
Compilers and Interpreters

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

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

Specific topics to be covered may include the following:
- Compiler architecture. Single-pass vs. multiple-pass translation.
- Lexical analysis (scanning). Design of scanners using finite automata. Tabular representations. Tools for building scanners.
- Parsing (syntax analysis). Top-down vs. bottom-up parsing. Parse trees and abstract syntax trees. Tabular representations for parsers. Parser generators.
- Symbol tables. Efficient algorithms and data structures. Representing data types in symbol tables.
- Type checking. Scope control. Static vs. dynamic type checking.
- Memory management. Static allocation, register allocation, stack allocation, heap allocation. Garbage collection.

- Code generation. Translating imperative programming constructs. Function and procedure calls. Branching code. Translating object-oriented constructs and modules.
- Optimization. Local and global optimizations. Dead code removal. Control flow analysis.

Prerequisites: general prerequisites; COSC 3301 3.0 recommended.
Degree Credit Exclusion: AK/COSC3420 6.0

COSC 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 synchronize with each other.
- Real time software. JSD modeling 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 analyzing 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, including COSC 3311 3.0 or COSC3221 3.0 or COSC3321 3.0.

COSC 4351 3.0 (x-listed COSC5341 3.0)
Real-Time Systems Theory
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 modeling, 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, including COSC 3311 3.0 or COSC3221 3.0 (or COSC3321 3.0) or COSC3341 3.0 (or COSC3111 3.0).

COSC 4352 3.0 (x-listed COSC5342 3.0)
Real-Time Systems Practice
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 a factory floor must stop the robot in time to prevent a collision. Other examples of real-time systems include communication systems, traffic systems, nuclear power plants and avionic systems. Real-time systems are complex and require a knowledge of reactive programs for their design. A reactive program maintains an ongoing non-terminating interaction with its environment rather than computing some final value on termination.

The course will focus on the design, construction and verification of soft and hard real-time systems. Topics may include:
- models of concurrent processes with access to a clock (e.g. by fair transition systems with timeouts and clock variables),

- semaphores and synchronization,
- process communication and fairness,
- temporal logic for specifying safety properties (e.g. freedom from deadlock),
- liveness and real-time response,
- verification of real-time systems using temporal logic model-checking tools such as StateClock/SteP
- examples from real-time programming languages (Ada and Java).

Prerequisites: general prerequisites, including COSC 3301 3.0 or COSC3311 3.0 or COSC3221 3.0 (or COSC3321 3.0).

COSC 4401 3.0 (x-listed COSC 5326 3.0)
Artificial Intelligence
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, abduction, explanation-based learning, learning k-DNF.
- Statistical learning: reinforcement learning, genetic learning algorithms, 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, including COSC 3402 3.0

COSC 4402 3.0 (x-listed COSC 5311 3.0)
Logic Programming
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 which these systems are designed to achieve. The most commonly recognized 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, including COSC 3401 3.0, and COSC3101 3.0 or COSC3341 3.0 (or COSC3111 3.0).

COSC 4411 3.0 (x-listed COSC5411 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 Readings:
- R. Elmasri and S.B. Navathe, *Fundamentals of Database Systems*, 2nd Ed., Benjamin Cummings, 1994.

Prerequisites: general prerequisites, SC/AS/COSC3421 3.0

COSC 4421 3.0 (x-listed COSC 5324 3.0)
Introduction to Robotics
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 sonars, several workstations and an extensive collection of software.

Prerequisites: general prerequisites; MATH1025 3.0

COSC 4422 3.0 (x-listed COSC 5323 3.0)
Computer Vision
Computer Vision is a very challenging problem with wide applications. It spans several disciplines within science and engineering: computer science, computer engineering, photogrammetry, telecommunications, robotics, medicine and the list goes on. This course introduces the fundamental concepts of vision with emphasis on computer science.

In particular the course covers the image formation process, color analysis, image processing, enhancement and restoration, feature extraction and matching, 3-D parameter estimation and applications. A Vision Laboratory is available where students can gain practical experience. The Lab includes several workstations equipped with video cameras, digitizers and image processing software.

Prerequisites: general prerequisites, including COSC3121 3.0

COSC 4431 3.0 (x-listed COSC 5331 3.0)
Computer Graphics

This course introduces the fundamental concepts of three-dimensional computer graphics. Algorithms for image generation and the components of interactive graphics systems are presented. The course discusses also virtual reality systems, how interactive entertainment systems, such as games, work and how animations for film and TV are created.

The first half of the course concentrates on the fundamentals of image generation: the graphics pipeline, modeling, graphics data structures, transformations, camera & perspective, visibility, raster graphics, shading.

The second part covers more advanced techniques and application areas: texturing, anti-aliasing, ray tracing, free-form surfaces, Interactive graphics systems, virtual reality, animation.

The assignments use current graphics toolkits to implement interactive graphics programs. Students work in small groups. Students are expected to be familiar with C and UNIX and will be using the X window environment on the undergraduate workstations.

Prerequisites: general prerequisites; MATH1025 3.0
Degree Credit Exclusion: COSC4331 3.0

COSC 4441 3.0
Human Computer Interaction

- 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, Color); Societal Impact of User Interfaces (Information Overload); Computer Supported Cooperative Work (CSCW, Synchronous and Asynchronous); Information Search and Visualization (Queries, Visualization, Data Mining)

The topics of this course will be applied in practical assignments and/or group projects. The projects will consists of a design part, an implementation part and user tests to evaluate the prototypes.

Suggested Reading:

- Alan Dix, Janet Finlay, Gregory Abowd, Russell Beale, Human-Computer Interaction, 2nd ed, Prentice Hall, 1998.

Prerequisites: general prerequisites; COSC/ITEC3461 3.0 recommended
Degree Credit Exclusions: COSC4341 3.0

COSC 4451 3.0 (x-listed COSC 5325 3.0)
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

Prerequisites: general prerequisites; COSC3121 3.0 or MATH3241 3.0.
Degree Credit Exclusions: COSC4242 3.0, EATS4020 3.0, MATH4130B 3.0, MATH4830 3.0, PHYS4060 3.0.

COSC 4461 3.0
Hypermedia and Multimedia Technology
The course focuses this year on the design and implementation of hypermedia presentation systems. "Hypermedia" refer to the non-linear organization of digital information, in which items (such as a word in a text field or a region of an image) are actively linked to other items. Users interactively select and traverse links in a hypermedia presentation system in order to locate specific information or entertainment, or to browse in large archives of text, sound, images, and video. Well-structured hypermedia give 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 markup languages;
- Networked hypermedia (e. g. HTTP browsers); performance and compression issues;
- Development Tool Kits;
- Distribution and Intellectual Property Issues.

Students will be expected to familiarize 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 (161 CCB), 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 which will be presented in class.

Prerequisites: general prerequisites, including ITEC/COSC3461 3.0.
Degree Credit Exclusion: COSC4361 3.0

Required Mathematics Courses

The introductory courses MATH1090 3.0, MATH1300 3.0, MATH1310 3.0, and MATH2090 3.0 are required of all Computer Science majors. Students who have not taken OAC calculus should consult advisors in the Mathematics Department to determine which courses they should take before attempting MATH1300 3.0. In addition some combination, or all of, the following courses are also required, depending on the degree program - MATH1025 3.0, and MATH2320 3.0.

Mathematics Substitute Course List

Course	***Acceptable Substitutions for COSC degree requirements***
MATH1025 3.0	MATH2021 3.0, MATH2221 3.0, MATH1021 3.0
MATH1300 3.0	MATH1000 3.0, MATH1013 3.0
MATH1310 3.0	MATH1010 3.0, MATH1014 3.0

Although not formally required for a computer science degree many other areas of mathematics are relevant to computer science. They include

- Probability and statistics: MATH1131 3.0, MATH1132 3.0, MATH2030 3.0, MATH2131 3.0, MATH3131 3.0, MATH3132 3.0
- (more) algebra: MATH2222 3.0, MATH2022 3.0, MATH3020 6.0, MATH3140 6.0
- combinatorics and graph theory: MATH2260 3.0, MATH3260 3.0

But in selecting mathematics courses please remember that most computer science honours degrees require you to take at least 30 credits that are not COSC and not MATH. The breadth of education implied by such a requirement is important for a computer science professional.

Messages for Atkinson College Students

Prerequisite and new-degree-requirement substitutes
Old Atkinson Computer Science courses will be accepted as prerequisites for courses in the combined AK/AS/SC/COSC course list according to the following table of prerequisite substitutes. The same substitutes will also be accepted in cases in which specific courses are required under the new degree requirements or for satisfying breadth requirements:

Current AK/AS/SC/COSC courses	substitute old AK/COSC
1020/1030	2411/2412
2001	3431
2011	3501
2021	3411
2031	no substitute
3101	3432
3111	no substitute
3121	3511
3122	3512
3201	no substitute
3211	3409A
3212	3409B
3301	no substitute
3311	no substitute
3321	3470
3331	3650
3401	3551
3402	3551
3408	3451
3418	4071
3421	3503
3461	no substitute
3530	no substitute
4101	no substitute
4111	4021

for current AK/AS/SC/MATH courses	substitute old AK/MATH
1090	2441
2320	2442

Old-degree-requirement substitutes

For continuing Atkinson Computer Science students who choose to graduate under the old degree requirements that were in effect when they were admitted to the Computer Science major, new AK/AS/SC/COSC courses will be accepted as substitutes for specific old degree requirements that have not yet been completed, according to the following table:

for old AK/COSC course	substitute current AK/AS/SC/COSC course
3411	2021
3431	2001
3460	3201
3501	2011
3502	any 32xx or 33xx course
3511	3121
3512	3122

for old AK/MATH course	Substitute current AK/AS/SC/MATH course
2441	1090
2442	2320

Upper Level Computer Science Requirements - All Faculties

Breadth requirement

Courses are classified into four areas at the 3000- and 4000-level as a means of guiding course selection by students.

The four areas are as follows:

- **Theory and Numerical Computation –** Course numbers COSC31xx 3.0, COSC41xx 3.0; topics: algorithms, data structures and complexity, automata and computability, program verification, scientific and numerical computing.
- **Systems –** Course numbers COSC32xx 3.0, COSC42xx 3.0; topics: digital logic, architecture, operating systems, data communication, and networks.
- **Software Development –** Course numbers COSC33xx 3.0, COSC43xx 3.0; topics: programming languages, software systems design and verification.
- **Applications –** Course numbers COSC34xx 3.0, COSC44xx 3.0; topics: artificial intelligence, expert systems, logic programming, databases, simulation, machine learning, robotics and computer vision.

All degree types require COSC3101 3.0, COSC3221 3.0, COSC3311 3.0 and at least one course from the Applications area at the 3000-level, thereby satisfying

breadth in computer science. The Specialised Honours degree requires one of COSC4101 3.0 or COSC4111 3.0 in addition.

Exceptions to Course Numbering
Service courses at all levels have the second digit 5. These courses do not satisfy requirements in Computer Science and grades will not be included in the Computer Science prerequisite grade point average calculation.

Other courses falling outside the course numbering conventions are the following.

- COSC3001 1.0 -- Organization and Management Seminar in Space and Communication Sciences
- COSC3002 1.0 -- Organization and Management Seminar
- COSC3010 3.0 -- Special Topics course
- COSC4001 6.0 -- restricted to SCS stream students
- COSC4010 3.0 -- Special Topics course
- COSC4080 3.0 -- Computer Science Project

Normal Order of Study

This section presents a summary of the Department's course requirements, by suggesting the normal order in which courses should be taken. There are also checklists for each program type at the back of this calendar.

Note: the Specialized Honours Space and Communication Sciences Stream has exceptions from the general requirements; the exceptions are noted. The course requirements of the SCS stream are described in the section on Program Checklists.

The indication of first year, second year, etc., indicates the year of study for normal progress by full-time students.

1000-level – first year

- Fall – COSC1020 3.0, MATH1090 3.0, MATH1300 3.0
- Winter – COSC1030 3.0, MATH1310 3.0.
- 15 additional credits satisfying general education, faculty, second major program, or elective requirements

2000-level – second year

- COSC2001 3.0, COSC2011 3.0, COSC2021 3.0, COSC2031 3.0
- *Specialized Honours:* MATH2090 3.0, MATH1025 3.0, MATH2320 3.0
  other Honours programs: MATH2090 3.0; MATH1025 3.0 **or** MATH2320 3.0
  BA and BSc (90-credit) programs: MATH209 3.0
- 9 to 15 additional credits satisfying general education, faculty, second major program, or elective requirements

3000-level – third year

- 12 COSC credits at the 3000-level satisfying the breadth requirement - COSC3101 3.0, COSC3221 3.0, COSC3311 3.0, plus COSC34xx 3.0
- 6 additional 3000-level COSC credits for the Bachelors and Specialized Honours programs
- 9 to 15 credits additional credits satisfying general education, Faculty, second major program, or elective requirements

4000-level – fourth year, honours programs only

- 12 COSC credits at the 4000-level (except for the Honours Minor BA degree which requires 6 credits at the 4000-level), including one of COSC4111 3.0 or COSC4101 3.0 for the Specialized Honours program (except SCS).
- 6 additional COSC credits at the 3000- or 4000-level for Specialized Honours programs (except SCS)
- 12 to 18 additional credits satisfying general education, Faculty, second major program, or elective requirements

Prerequisites for Computer Science Courses[1]

It is essential that students fulfill prerequisites for courses they wish to take.

There are both **general** prerequisites which are required for all COSC courses at the specified level and **specific** prerequisites for each course which are in addition to the general prerequisites. Both types of prerequisites include computer science courses and mathematics courses, and in all cases there are grade requirements in the prerequisite courses. The prerequisites are listed after each course description and summarized in the following table.

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

Course Title		Prerequisite(s)
1000-Level		
COSC1020 3.0	Intro. to Computer Science I	Refer to course description
COSC1030 3.0	Intro. to Computer Science II	COSC1020 3.0

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

2000-Level

General Prerequisites:

- completed COSC1030 3.0
- completed MATH1090 3.0
- a cumulative GPA of 4.5 or better for completed Computer Science courses.

COSC2001 3.0	Intro. to Theory of Computation	General prerequisites
COSC2011 3.0	Fundamentals of Data Structures	General prerequisites
COSC2021 3.0	Computer Organization	General prerequisites
COSC2031 3.0	Software Tools	General prerequisites

3000-Level

General Prerequisites (except 35xx x.x courses)

- completed COSC2011 3.0, and one of COSC2001 3.0 or COSC2021 3.0 or COSC2031 3.0
- completed MATH1300 3.0 and MATH1310 3.0
- completed one of MATH2090 3.0, MATH1025 3.0 or MATH2320 3.0
- a cumulative GPA of 4.5 or better over all completed Computer Science courses.

Theory and Numerical Computation	**Prerequisites**
COSC3101 3.0 Design and Analysis of Algorithms	General prerequisites including MATH2090 3.0 or MATH2320 3.0
COSC3121 3.0 Intro. to Numerical Computations I	One of COSC1540 3.0, COSC2031 3.0 or COSC3501 3.0; one of MATH1010 3.0, MATH1310 3.0 or MATH1014 3.0; one of MATH1021 3.0, MATH1025 3.0, MATH2021 3.0 or MATH2221 3.0
COSC3122 3.0 Intro. to Numerical Computations II	COSC3121 3.0; MATH2270 3.0

Systems	
COSC3201 3.0 Digital Logic Design	General prerequisites including COSC2021 3.0
COSC3211 3.0 Data Communication	General prerequisites including COSC2021 3.0; MATH2090 3.0
COSC3213 3.0 Computer Networks	General prerequisites
COSC3221 3.0 Operating System Fundamentals	General prerequisites including COSC2021 3.0; COSC2031 3.0

Software Development

COSC3301 3.0 Programming Language Fundamentals	General prerequisites including COSC2001 3.0
COSC3311 3.0 Software Design	General prerequisites including COSC2001 3.0; COSC 2031 3.0, MATH2090 3.0
COSC3331 3.0 Object-oriented Programming and Design	General prerequisites
COSC3341 3.0 Intro. to Program Verification	General prerequisites including MATH2090 3.0

Applications

COSC3401 3.0 Introduction to Symbolic Computation	General prerequisites including MATH2090 3.0
COSC3402 3.0 Intro. to Concepts of Artificial Intelligence	COSC3401 3.0; MATH2090 3.0 or MATH2320 3.0
COSC3408 3.0 Simulation of Discrete Systems	General prerequisites; MATH2560 3.0
COSC3418 3.0 Simulation of Continuous Systems	General prerequisites; MATH2560 3.0
COSC3421 3.0 Introduction to Database Systems	ITEC/COSC2011 3.0
COSC3461 3.0 User Interfaces	ITEC/COSC2011 3.0 or COSC2031 3.0

Other Courses:

COSC3001 1.0 Org. & Management Seminar in SCS	In 3rd year of SCS stream
COSC3010 3.0 Special Topics in Computer Science	Varies depending on the topic

4000-Level

General Prerequisites:

- completed COSC2001 3.0; COSC2011 3.0; COSC2021 3.0; COSC2031 3.0
- completed MATH2090 3.0
- completed at least 12 credits in computer science 3000-level courses.
- a cumulative GPA of 4.5 or better over all completed computer science courses

Theory Courses	Specific Prerequisites
COSC4101 3.0 Advanced Data Structures	COSC3101 3.0
COSC4111 3.0 Automata and Computability	COSC3101 3.0

Systems Courses

COSC4201 3.0 Computer Architecture	COSC3201 3.0; COSC3221 3.0
COSC4211 3.0 Performance Evaluation of Computer Systems	COSC3211 3.0 or COSC3213 3.0; COSC3408 3.0
COSC4213 3.0 Computer Networks II	COSC3213 3.0
COSC4221 3.0 Operating System Design	COSC3221 3.0

Software Courses

COSC4301 3.0 Programming Language Design	COSC3301 3.0
COSC4302 3.0 Compilers and Interpreters	(COSC3301 3.0 recommended)
COSC4311 3.0 System Development	COSC3311 3.0 or COSC3221 3.0
COSC4351 3.0 Real-Time Systems Theory	COSC3341 3.0 or COSC3311 3.0 or COSC3221 3.0
COSC4352 3.0 Real-Time Systems Practice	COSC3301 3.0 or COSC3311 3.0 or COSC3221 3.0

Applications Courses

COSC4401 3.0 Artificial Intelligence	COSC3402 3.0
COSC4402 3.0 Logic Programming	COSC3401 3.0; COSC3101 3.0 or COSC3341 3.0
COSC4411 3.0 Database Management Systems	COSC3421 3.0
COSC4421 3.0 Introduction to Robotics	MATH1025 3.0
COSC4422 3.0 Computer Vision	COSC3121 3.0 (MATH3241 3.0)
COSC4431 3.0 Computer Graphics	MATH1025 3.0
COSC4441 3.0 Human Computer Interaction	COSC3461 3.0
COSC4451 3.0 Signals and Systems	COSC3121 3.0 (MATH3241 3.0)
COSC4461 3.0 Hypermedia and Multimedia Technologies	COSC3461 3.0

Other Courses

COSC4001 6.0Space and Comm. Sciences Workshop	3000-level of SCS core
COSC4080 3.0 Computer Science Project	permission of course director, 36 COSC credits
COSC4010 3.0 Special Topics in Computer Science	Varies depending on the topic

FPAS 2001-02 Checklist[1] — BSc Degree

Computer Science Requirements — Credit Count

1000-level	COSC1020 3.0	MATH1090 3.0	MATH1300 3.0		9
	COSC1030 3.0	MATH1310 3.0			6
2000-level	COSC2001 3.0	COSC2011 3.0	COSC2021 3.0	COSC2031 3.0	12
	MATH2090 3.0				3

3000-level One course from each area:

Theory	COSC3101 3 0	Software	COSC3311 3.0	6
Systems	COSC3221 3.0	Applications	COSC34____ 3.0	6
Two more courses	COSC3_____ 3.0		COSC3_____ 3.0	6

Faculty Requirements

General Education Courses: _____ _____ 12

6 credits from:

BIOL1010 6.0	BIOL1410 6.0	CHEM1000 3.0
CHEM1001 3.0	EATS1010 3.0	EATS1011 3.0
PHYS1010 6.0	PHYS1410 6.0	6

At least 3 additional credits from 1000-level Science courses (excluding CHEM1500 4.0, MATH1510 6.0, MATH1515 3.0, PHYS1510 4.0 and all Natural Science courses) 3

Additional courses totaling 21 credits and satisfying
1. more SC credits (as required for a total of 66)

_____ _____ _____ _____

_____ _____ _____ 21

Total credits 90

---

[1] *A minimum cumulative grade point average of 4.0 over all courses is required to graduate. In addition, the Departmental prerequisite GPA over COSC courses must be met to proceed in the program.*

FPAS 2001-02 Checklist[1] BSc Hons Double Major Degree
BSc Hons. Major/Minor (COSC Major) Degree

Computer Science Requirements					**Credit Count**
1000-level	COSC1020 3.0	MATH1090 3.0	MATH1300 3.0		9
	COSC1030 3.0	MATH1310 3.0			6
2000-level	COSC2001 3.0	COSC2011 3.0	COSC2021 3.0	COSC2031 3.0	12
	MATH2090 3.0	MATH1025 3.0 **or** MATH2320 3.0			6
3000-level	One course from each area				
	Theory COSC3101 3.0	Software COSC3311 3.0			6
	Systems COSC3221 3.0	Applications COSC34____ 3.0			6
4000-level	Four courses	COSC4_____ 3.0	COSC4_____ 3.0		6
		COSC4_____ 3.0	COSC4_____ 3.0		6

Faculty Requirements[2]

General Education courses:	_____	_____		12
6 credits from:	BIOL1010 6.0	BIOL1410 6.0	CHEM1000 3.0	
	CHEM1001 3.0	EATS1010 3.0	EATS1011 3.0	
	PHYS1010 6.0	PHYS1410 6.0		6
At least 3 additional credits from 1000-level Science courses (excluding CHEM1500 4.0, MATH1510 6.0, MATH1515 3.0, PHYS1510 4.0 and all Natural Science courses)				3

Other Honours Subject and Other Courses (total 42 more credits)

Including
1. non-COSC/non-MATH credit for a total of 30,
2. additional 3000- and 4000-level credit for a total of 42
3. additional SC credit for a total of 90

_____	_____	_____	_____	_____	15
_____	_____	_____	_____	_____	15
_____	_____	_____	_____		12
				Total credits	**120**

---

1 *A minimum cumulative grade-point-average of 5.0 over all courses is required to proceed in each year of the program and to graduate. If the second major is BIOL a minimum cumulative grade-point-average of 6.0 over all SC courses is also required. In addition, the Departmental prerequisite GPA over COSC courses must be met to proceed in the program.*

2 *The other major may include additional general education and 1000-level SC requirements.*

FPAS 2001-02 Checklist[1]

BSc Honours Major/Minor (COSC Minor) Degree

Computer Science (Minor) Requirements					**Credit Count**
1000-level	COSC1020 3.0	MATH1090 3.0	MATH1300 3.0		9
	COSC1030 3.0	MATH1310 3.0			6
2000-level	COSC2001 3.0	COSC2011 3.0	COSC2021 3.0	COSC2031 3.0	12
	MATH2090 3.0	MATH1025 3.0 **or** MATH2320 3.0			6
3000-level	One course from each area				
	Theory	COSC3101 3.0	Software	COSC3311 3.0	6
	Systems	COSC3221 3.0	Applications	COSC34____ 3.0	6
4000-level	Two half courses:	COSC4_____ 3.0		COSC4_____ 3.0	6

Faculty Requirements[3]

General Education courses:	_____	_____		12
6 credits from:	BIOL1010 6.0	BIOL1410 6.0	CHEM1000 3.0	
	CHEM1001 3.0	EATS1010 3.0	EATS1011 3.0	
	PHYS1010 6.0	PHYS1410 6.0		6
At least 3 additional credits from 1000-level Science courses (excluding CHEM1500 4.0, MATH1510 6.0, MATH1515 3.0, PHYS1510 4.0 and all Natural Science courses)				3

Other Honours Subject and Other Courses (total 48 more credits)

Including
1. non-COSC/non-MATH credit for a total of 30,
2. additional 3000- and 4000-level credit for a total of 42
3. additional SC credit for a total of 90

_____	_____	_____	_____	12
_____	_____	_____	_____	12
_____	_____	_____	_____	12
_____	_____	_____	_____	12
			Total credits	**120**

---

1 *A minimum cumulative grade-point-average of 5.0 over all courses is required to proceed in each year of the program and to graduate. If the major is BIOL a minimum cumulative grade-point-average of 6.0 over all SC courses is also required. In addition, the Departmental prerequisite GPA over COSC courses must be met to proceed in the program.*

3 *The other major may include additional general education and 1000-level SC requirements.*

FPAS 2001-02 Checklist[1] BSc Specialized Honours Degree

Computer Science Requirements					**Credit Count**
1000-level:	COSC1020 3.0	MATH1090 3.0	MATH1300 3.0		9
	COSC1030 3.0	MATH1310 3.0			6
2000-level:	COSC2001 3.0	COSC2011 3.0	COSC2021 3.0	COSC2031 3.0	12
	MATH2090 3.0	MATH1025 3.0	MATH2320 3.0		9
3000-level:	One course from each area				
	Theory	COSC3101 3.0	Software	COSC3311 3.0	6
	Systems	COSC3221 3.0	Applications	COSC34____ 3.0	6
	Two more courses:				
		COSC3_____ 3.0	COSC3_____ 3.0		6
4000-level:	COSC4101 3.0 **or**	COSC4111 3.0			3
	COSC4_____ 3.0	COSC4_____ 3.0	COSC4_____ 3.0		9
Two courses (3000- or 4000-level)					
		COSC_____ 3.0	COSC_____ 3.0		6

Faculty Requirements					
General Education Courses	_____	_____			12
6 credits from:	BIOL1010 6.0	BIOL1410 6.0	CHEM1000 3.0		
	CHEM1001 3.0	EATS1010 3.0	EATS1011 3.0		
	PHYS1010 6.0	PHYS1410 6.0			6
At least 3 additional credits from 1000-level Science courses (excluding CHEM1500 4.0, MATH1510 6.0, MATH1515 3.0, PHYS1510 4.0 and all Natural Science courses)					3
Additional courses totalling 27 credits and satisfying 1. More SC credits (as required for a total of 90) 2. More non-COSC, non-MATH credits (as required for a total of 30) 3. More 3000- or 4000-level credit (as required for a total of 42)					
_____	_____	_____	_____	_____	15
_____	_____	_____	_____		27
				Total credits	**120**

---

[1] *A minimum cumulative grade-point-average of 5.0 over all courses is required to proceed in each year of the program and to graduate. In addition, the Departmental prerequisite GPA over COSC courses must be met to proceed in the program.*

FPAS 2001-02BSc Specialized Honours Degree, SCS Stream[1]

Computer Science Requirements			Credit Count
1000-level		COSC1020 3.0 MATH1090 3.0 MATH1013 3.0	9
		COSC1030 3.0 MATH1014 3.0 MATH1025 3.0	9
		PHYS1010 6.0 CHEM1000 6.0 **or** EATS1010 6.0	12
2000-level		COSC2001 3.0 COSC2011 3.0 COSC2021 3.0	9
		MATH2015 3.0 MATH2090 3.0 MATH2270 3.0	9
		PHYS2020 3.0 PHYS2040 3.0 PHYS2211 1.0	7
	One of	PHYS2010 3.0 **or** EATS2470 4.0	3 or 4
	One of	CHEM2011 3.0 **or** COSC2031 3.0 **or** EATS2010 3.0 **or** EATS2030 3.0 **or** PHYS1070 3.0 **or** PHYS2060 3.0	3
3000-level		COSC3121 3.0 COSC3211 3.0 COSC3221 3.0	9
		COSC/EATS/PHYS/3001 1.0 EATS/PHYS3280 3.0	4
		PHYS3050 3.0 PHYS3250 3.0	6
	One of	COSC3201 3.0 **or** COSC3212 3.0 **or** COSC3311 3.0 **or** COSC3331 3.0	3
	One of	any 3000-level COSC course not already taken (without second digit 5) **or** EATS3020 3.0 **or** EATS3030 3.0 **or** MATH3271 3.0 **or** MATH3410 3.0 **or** PHYS3020 3.0 **or** PHYS3070 3.0 **or** PHYS3080 3.0 **or** PHYS3150 3.0 **or** PHYS4120 3.0 **or** other approved courses	3
4000-level		COSC4001 6.0	6
	One of	COSC4201 3.0 **or** COSC4351 3.0 **or** COSC4352 3.0	3
	One of	COSC4301 3.0 **or** COSC4302 3.0 **or** COSC4321 3.0 **or** COSC4341 3.0	3
	Two of	COSC4242 3.0 **or** COSC4331 3.0 **or** COSC4421 3.0 **or** COSC4422 3.0	6
	Two of	4000-level COSC courses not already taken as listed above **or** EATS4220 3.0 **or** EATS4230 3.0 **or** PHYS3070 3.0 **or** PHYS4060 3.0 **or** PHYS4110 3.0 **or** PHYS4270 4.0 **or** PHYS4450 3.0	6
Faculty Requirements			
General Education Courses		_____ _____	12
		Total credits 122 or 123	

[1] *A minimum cumulative grade-point-average of 6.0 over all SC courses and 5.0 over ALL courses is required to proceed in each year of the program and to graduate. In addition, the Departmental prerequisite GPA over COSC courses must be met to proceed in the program.*

Computer Science Requirements
Credit Count

1000-level	COSC1020 3.0	MATH1090 3.0	MATH1300 3.0		9
	COSC1030 3.0	MATH1310 3.0			6
2000-level	COSC2001 3.0	COSC2011 3.0	COSC2021 3.0	COSC2031 3.0	12
	MATH2090 3.0				3

3000-level One course from each area

Theory	COSC3101 3.0	Software	COSC3311 3.0	6
Systems	COSC3221 3.0	Applications	COSC34____ 3.0	6

Two more half courses:
COSC3_____ 3.0 COSC3_____ 3.0 6

Faculty Requirements

General education

1000-level: NATS_____ 6.0 6

One of HUMA_____ 9.0 **or** SOSC_____ 9.0 9

2000-level:
(must be HUMA if a 1000-level SOSC was chosen; or SOSC if a 1000-level HUMA was chosen)

One of HUMA_____ 9.0 **or** SOSC_____ 9.0 9

Electives 18 credits outside COSC requirements

_____ _____ _____ 9

_____ _____ _____ 9

Total credits 90

---

1 *A cumulative grade point average of 4.0 over all courses is required to graduate. In addition, the Departmental prerequisite GPA over COSC courses must be met to proceed in the program.*

Computer Science Requirements

Credit Count

1000-level	COSC1020 3.0	MATH1090 3.0	MATH1300 3.0		9
	COSC1030 3.0	MATH1310 3.0			6
2000-level	COSC2001 3.0	COSC2011 3.0	COSC2021 3.0	COSC2031 3.0	12
	MATH2090 3.0	MATH1025 3.0 **or**	MATH2320 3.0		6
3000-level	One course from each area				
	Theory	COSC3101 3.0	Software	COSC3311 3.0	6
	Systems	COSC3221 3.0	Applications	COSC34_____ 3.0	6
4000-level	Four half courses	COSC4_____ 3.0	COSC4_____ 3.0		6
		COSC4_____ 3.0	COSC4_____ 3.0		6

Faculty Requirements

General education

1000-level NATS_____ 6.0 — 6

One of HUMA_____ 9.0 **or** SOSC_____ 9.0 — 9

2000-level
(must be HUMA if a 1000-level SOSC was chosen; or SOSC if a 1000-level HUMA was chosen)

One of HUMA_____ 9.0 **or** SOSC_____9.0 — 9

Electives: 18 credits outside COSC requirements

_____ _____ _____ — 18

Additional courses: totaling 21 credits and satisfying

1. More 4000-level credits (as required for a total of 18)
2. More 3000- or 4000-level credits (as required for a total of 36)

_____ _____ _____ _____ — 12

_____ _____ _____ — 9

Total credits 120

---

[1] *A cumulative grade-point-average of 5.0 over all courses is required to proceed in each year of the program. In addition, the Departmental prerequisite GPA over COSC courses must be met to proceed in the program. To graduate requires a cumulative grade-point-average of 5.0 over all courses.*

Computer Science Requirements

Credit Count

1000-level	COSC1020 3.0	MATH1090 3.0	MATH1300 3.0		9
	COSC1030 3.0	MATH1310 3.0			6
2000-level	COSC2001 3.0	COSC2011 3.0	COSC2021 3.0	COSC2031 3.0	12
	MATH2090 3.0	MATH1025 3.0 **or**	MATH2320 3.0		6
3000-level	One course from each area				
	Theory	COSC3101 3.0	Software	COSC3311 3.0	6
	Systems	COSC3221 3.0	Applications	COSC34_____ 3.0	6
4000-level	Two half courses	COSC4_____ 3.0	COSC4_____ 3.0		6

Faculty Requirements

General education

1000-level

NATS_____ 6.0 — 6

One of HUMA_____ 9.0 **or** SOSC_____ 9.0 — 9

2000-level

(must be HUMA if a 1000-level SOSC was chosen; or SOSC if a 1000-level HUMA was chosen)

One of HUMA_____ 9.0 **or** SOSC_____9.0 — 9

Honours Major subject and other courses

(To satisfy requirements of the honours major, and upper-level requirements.)

_____	_____	_____	_____	12
_____	_____	_____	_____	12
_____	_____	_____	_____	12
_____	_____	_____		9

Total credits 120

---

[1] *A cumulative grade-point-average of 5.0 over all courses is required to proceed in each year of the program. In addition, the Departmental prerequisite GPA over COSC courses must be met to proceed in the program. To graduate requires a cumulative grade-point-average of 5.0 over all courses.*

Faculty of Arts 2001-02 Checklist[1] BA Specialized Honours Degree

Computer Science Requirements
Credit Count

1000-level	COSC1020 3.0	MATH1090 3.0	MATH1300 3.0		9
	COSC1030 3.0	MATH1310 3.0			6
2000-level	COSC2001 3.0	COSC2011 3.0	COSC2021 3.0	COSC2031	12
	MATH2090 3.0	MATH1025 3.0	MATH2320 3.0		9

3000-level One course from each area

Theory	COSC3101 3.0	Software	COSC3311 3.0	6
Systems	COSC3221 3.0	Applications	COSC34_____ 3.0	6

Two more courses:
COSC3_____ 3.0 COSC3_____ 3.0 6

4000-level	Four courses COSC4101 3.0 **or** COSC4111 3.0			3
	COSC4_____ 3.0	COSC4_____ 3.0	COSC4_____ 3.0	9

Two courses (3000- or 4000-level)
COSC_____ 3.0 COSC_____ 3.0 6

Faculty Requirements

General education

1000-level	NATS_____ 6.0			6
One of	HUMA_____ 9.0	**or**	SOSC_____ 9.0	9

2000-level
(must be HUMA if a 1000-level SOSC was chosen; or SOSC if a 1000-level HUMA was chosen)

One of	HUMA_____ 9.0	**or**	SOSC_____ 9.0	9

Electives (3 courses outside COSC requirements)

_____ _____ _____ 18

Additional courses: totaling 6 credits and satisfying
1. More 4000-level credits (as required for a total of 18)

_____ _____ 6

Total credits 120

---

[1] *A cumulative grade-point-average of 5.0 over all courses is required to proceed in each year of the program. In addition, the Departmental prerequisite GPA over COSC courses must be met to proceed in the program. To graduate requires a cumulative grade-point-average of 5.0 over all courses.*

Faculty of Arts 2001-02 Checklist[1]

BA Honours Double Major Degree

Computer Science Requirements
Credit Count

1000-level	COSC1020 3.0	MATH1090 3.0	MATH1300 3.0		9
	COSC1030 3.0	MATH1310 3.0			6
2000-level	COSC2001 3.0	COSC2011 3.0	COSC2021 3.0	COSC2031 3.0	12
	MATH2090 3.0	MATH1025 3.0 **or** MATH2320 3.0			6
3000-level	One course from each area				
	Theory	COSC3101 3.0	Software	COSC3311 3.0	6
	Systems	COSC3221 3.0	Applications	COSC34_____ 3.0	6
4000-level	Four half courses	COSC4_____ 3.0	COSC4_____ 3.0		6
		COSC4_____ 3.0	COSC4_____ 3.0		6

Faculty Requirements

General education

1000-level NATS_____ 6.0 — 6

One of HUMA_____ 9.0 **or** SOSC_____ 9.0 — 9

2000-level
(must be HUMA if a 1000-level SOSC was chosen; or SOSC if a 1000-level HUMA was chosen)

One of HUMA_____ 9.0 **or** SOSC_____ 9.0 — 9

Other Honours Major Subject and Other Courses
(To satisfy requirements of the other honours major, and upper-level requirements.)

_____ _____ _____ _____ — 12

_____ _____ _____ _____ — 12

_____ _____ _____ _____ — 12

_____ — 3

Total credits 120

---

[1] *A cumulative grade-point-average of 5.0 over all courses is required to proceed in each year of the program. In addition, the Departmental prerequisite GPA over COSC courses must be met to proceed in the program. To graduate requires a cumulative grade-point-average of 5.0 over all courses.*

Computer Science Requirements — Credit Count

1000-level	COSC1020 3.0	MATH1090 3.0	MATH1300 3.0	9
	COSC1030 3.0	MATH1310 3.0		6
2000-level	COSC2001 3.0	COSC2011 3.0	COSC2021 3.0 COSC2031 3.0	12
	MATH2090 3.0			3
3000-level	One course from each area			
	Theory	COSC3101 3 0	Software COSC3311 3.0	6
	Systems	COSC3221 3.0	Applications COSC34____ 3.0	6
	Two more courses	COSC3_____ 3.0	COSC3_____ 3.0	6

Faculty Requirements

General education (24 credits)

1000-level:	MATH1710 6.0 **or** MATH17____ 6.0 **or** MODES_____ 6.0	6
	HUMA_____ 6.0 SOSC_____ 6.0	12

6 credits from:

BIOL1010 6.0	BIOL1410 6.0	CHEM1000 3.0
CHEM1001 3.0	EATS1010 3.0	EATS1011 3.0
PHYS1010 6.0	PHYS1410 6.0	6

At least 3 additional credits from 1000-level Science courses (excluding CHEM1500 4.0, MATH1510 6.0, MATH1515 3.0, PHYS1510 4.0 and all Natural Science courses) 3

Electives: 1. 6 credits in Science (courses cross listed as SC) at the 2000-level or above
2. 9 additional credits

_____	_____	_____	9
_____	_____		6

Total credits 90

---

1 *A cumulative grade point average of 4.0 over all courses is required to proceed in each year of the program and to graduate. In addition, the Departmental general prerequisite cumulative grade point average over all COSC courses must be met to proceed in the program.*

Atkinson Faculty 2001-02 Checklist[1] BSc Honours Degree

Computer Science Requirements — Credit Count

1000-level: COSC1020 3.0 MATH1090 3.0 MATH1300 3.0 — 9
COSC1030 3.0 MATH1310 3.0 — 6

2000-level: COSC2001 3.0 COSC2011 3.0 COSC2021 3.0 COSC2031 3.0 — 12
MATH2090 3.0 MATH1025 3.0 MATH2320 3.0 — 9

3000-level: One course from each area

Theory COSC3101 3.0 Software COSC3311 3.0 — 6

Systems COSC3221 3.0 Applications COSC34____ 3.0 — 6

Two more courses:

COSC3_____ 3.0 COSC3_____ 3.0 — 6

4000-level: COSC4101 3.0 **or** COSC4111 3.0 — 3

COSC4_____ 3.0 COSC4_____ 3.0 COSC4_____ 3.0 — 9

Two courses (3000- or 4000-level)

COSC_____ 3.0 COSC_____ 3.0 — 6

Faculty Requirements

General education (24 credits)

1000-level: MATH1710 6.0 **or** MATH17____ 6.0 **or** MODES_____ 6.0 — 6

HUMA_____ 6.0 SOSC_____ 6.0 — 12

6 credits from:

BIOL1010 6.0	BIOL1410 6.0	CHEM1000 3.0
CHEM1001 3.0	EATS1010 3.0	EATS1011 3.0
PHYS1010 6.0	PHYS1410 6.0	

— 6

At least 3 additional credits from 1000-level Science courses (excluding CHEM1500 4.0, MATH1510 6.0, MATH1515 3.0, PHYS1510 4.0 and all Natural Science courses) — 3

Electives:
1. 6 credits in Science (courses cross listed as SC) at the 2000-level or above
2. 3 credits at the 3000-level or above (as required for a total of 39)
3. more non-COSC, non-MATH credits (as required for a total of 30)
4. more credits (as required for a total of 120)

_____ _____ _____ _____ — 12

_____ _____ _____ — 9

Total credits 120

---

1 *A cumulative grade-point-average of 5.0 over all courses is required to proceed in each year of the program and to graduate. In addition, the Departmental prerequisite GPA over COSC courses must be met to proceed in the program.*

Atkinson Faculty 2001-02 Checklist[1] BA Degree

Computer Science Requirements

Credit Count

1000-level	COSC1020 3.0	MATH1090 3.0	MATH1300 3.0	9
	COSC1030 3.0	MATH1310 3.0		6
2000-level:	COSC2001 3.0	COSC2011 3.0	COSC2021 3.0 COSC2031 3.0	12
	MATH2090 3.0			3
3000-level	One course from each area			
	Theory	COSC3101 3 0	Software COSC3311 3.0	6
	Systems	COSC3221 3.0	Applications COSC34____ 3.0	6
	Two more courses	COSC3_____ 3.0	COSC3_____ 3.0	6

Faculty Requirements

General education (24 credits)			
1000-level:	MATH1710 6.0 **or** MATH17____ 6.0 **or** MODES_____ 6.0		6
	HUMA_____ 6.0	SOSC_____ 6.0	12
	NATS _____ 6.0		6

Electives: 1. 6 credits outside COSC requirements
2. 6 credits at the 3000-level or above
3. 6 credits anything

_____	_____	_____	9
_____	_____	_____	9

Total credits 90

---

1 *A cumulative grade point average of 4.0 over all courses is required to graduate. In addition, the Departmental general prerequisite cumulative grade point average over all COSC courses must be met to proceed in the program.*

Atkinson Faculty 2001-02 Checklist[1]

BA Specialized Honours Degree

Computer Science Requirements					**Credit Count**
1000-level:	COSC1020 3.0	MATH1090 3.0	MATH1300 3.0		9
	COSC1030 3.0	MATH1310 3.0			6
2000-level:	COSC2001 3.0	COSC2011 3.0	COSC2021 3.0	COSC2031 3.0	12
	MATH2090 3.0	MATH1025 3.0	MATH2320 3.0		9
3000-level:	One course from each area				
	Theory	COSC3101 3.0	Software	COSC3311 3.0	6
	Systems	COSC3221 3.0	Applications	COSC34____ 3.0	6
	Two more courses:				
		COSC3_____ 3.0	COSC3_____ 3.0		6
4000-level:	COSC4101 3.0 **or** COSC4111 3.0				3
	COSC4_____ 3.0	COSC4_____ 3.0	COSC4_____ 3.0		9
Two courses (3000- or 4000-level)					
		COSC_____ 3.0	COSC_____ 3.0		6

Faculty Requirements

General education (24 credits)				
1000-level:	MATH1710 6.0 **or** MATH17____ 6.0 **or** MODES_____ 6.0			6
	HUMA_____ 6.0		SOSC_____ 6.0	12
	NATS _____ 6.0			6

Electives: 1. 9 credits outside COSC requirements at 3000-level or above (or if MATH at the 2000-level or above)
2. more credits outside COSC and MATH (as required for a total of 30 credits)
3. more credits (as required for a total of 120)

_____	_____	_____	_____	12
_____	_____	_____	_____	12
			Total credits	**120**

[1] *A cumulative grade-point-average of 5.0 over all courses is required to proceed in each year of the program. In addition, the Departmental general prerequisite cumulative grade-point-average over all Computer Science courses must be met to proceed in the program. To graduate requires a cumulative grade-point-average of 5.0 over all courses.*