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 practise it. We recommend a maximum of three Computer Science courses per term.

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

Peter Cribb, Undergraduate Director — 125 CCB — Tel. (416)736-5334
Email: enquiries@cs.yorku.ca

John Amanatides, Graduate Program Director — 125 CCB — Tel. (416)736-5334
www.cs.yorku.ca/grad/

Michael Jenkin, Chair — 126 CCB — Tel. (416)736-5053
Fax (416)736-5872

CSAC Accreditation

All Computer Science honours programmes offered in the Faculty of Pure and Applied Science and the Faculty of Arts, with the exception of the BA honours minor, are accredited by the Computer Science Accreditation Council (CSAC).

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 programmes worthy of recognition. The objectives of the Council are:

- to formulate and maintain high educational standards for Canadian universities offering computer and information science programmes, and to assist those institutions in planning and carrying out education programmes.
- 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 Programme simplifies the process of professional certification as an Information Systems Provider 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.

Admission to the Computer Science Major

The educational background of students who seek admission to a Computer Science degree programme 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 either the Faculty of Arts (for the BA degree) or the Faculty of Pure and Applied Science (for the B.Sc. degree). Each Faculty has certain admission requirements that must be met. The requirements described below are in addition to these faculty admission (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 all mathematics credits (i.e. an average over all 3 if 3 OACs were taken) 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 programme 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

Voice Response Enrolment System

Students enrol in courses using the York University Voice Response Enrolment System, 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.

Waiting Lists

Full courses are removed from VRES and students are invited to apply via a waiting list form. Waiting list applications are accepted up until the middle of the first week of each term. A waiting list application does not constitute enrolment in a course. The student must check with the Department as to the result of their application and use a VRES Special Permission Window assigned to them if their application is successful.

Processing Waiting Lists

Waiting list applications are **not** treated on a first-come first-served basis. Decisions are made according to the following criteria:

- normal progress through the degree (i.e. 1000-level courses in year one, 2000-level in 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.
- the closer to graduation, the higher the priority.

Limits on Course Enrolment

A maximum of two 2000-level Computer Science courses in one term is permitted. Three 3000- or 4000-level courses per term is normal. Specialized honours students may take four upper level computer science courses. In the summer term students are not permitted to take more than 6 credits in Computer Science. Under no circumstances will students be permitted to take five computer science courses in one term. If any student enrols in more than four upper year or two 2000-level computer science 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 Atkinson College or at another institution should either consult the Atkinson's Course Equivalent/Exclusion tables or submit a Letter of Permission (LOP) form. The Atkinson's Course Equivalent/Exclusion table is available at the Office of Science Academic Services and published in the university's Undergraduate Lecture Schedule.

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.

Core Courses:

Core courses include all 1000- and 2000-level Computer Science courses, the 3000-level computer science courses satisfying the breadth requirement, and any required 3000- and 4000-level computer science courses, ie. COSC3101 3.0, COSC4101 3.0, COSC4111 3.0.

Recent & Current (99/00) Academic Changes

1. **GPA Prerequisite in Computer Science**
   The Computer Science GPA prerequisite for 2000-level and higher Computer Science courses is 4.5 (for both Ordinary and Honours programme). **Note: This is a prerequisite for continuing to take Computer Science courses and not a graduation requirement.**

2. **Graduation Requirements (This is a new requirement for BSc. degree only - previously existed for BA)**
   Students admitted to York University for 1999/2000 and subsequent years, the Senate of York University will require a minimum overall grade-point average of 4.0 in order to be eligible to graduate in an undergraduate Ordinary degree programme.
   Students admitted to York University for 1999/2000 and subsequent years, the Senate of York University will require a minimum overall grade-point average of 5.0 in order to be eligible to graduate in an undergraduate Honours degree programme.

3. **Course Deletions and Additions**

   · COSC3411 3.0, File Structures and Data Management, will no longer be offered.

   · COSC3412 3.0, Introduction to Database Management Systems is replaced by COSC3421 3.0, Introduction to Database Systems.

   · COSC3461 3.0, Human Computer Interaction, is offered for the first time.

4. The BA (Ordinary) and B.Sc. (Ordinary) require MATH2320 3.0 rather than MATH2090 3.0

5. **Non-COSC, non-MATH courses**
   All Honours programmes, with the exception of the BA minor, require completion of at least 30 credits that are **not COSC *and* not MATH**.

6. **Organisation and Management Seminar Course**

There is a new course COSC3002 1.0, Organisation and Management Seminar. All students are encouraged to take this course since it provides important insight into the social responsibilities of a Computer Science professional.

7. New Prerequisites to Some Upper Level Courses.

COSC3311 3.0, COSC3321 3.0 and COSC4302 3.0 require COSC2031 3.0 as a prerequisite since programming in C or C++ is an important part of these courses.

Programmes 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 programme leading to an Honours (four year) degree in either Arts (B.A.) or Pure and Applied Science (B.Sc.). It may also be combined with most subjects in both Arts and Science leading to a four-year double major degree (B.A. or B.Sc.).

The recommended courses in computer science and mathematics are identical in most programmes in the first two years of study so that students can make their final decisions as to which programme to graduate in after they have more exposure to the discipline.

Ordinary vs. Honours

An Ordinary programme requires 90 credits (normally completed in three years of study) and a grade point average of 4.0 over all courses (B.A. and B.Sc.). An honours programme 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 Ordinary degree.

Both Arts and Science programmes are structured in such a way that a student who embarks on an honours programme can meet the requirements for an Ordinary degree by the end of the third year and can at that time graduate with either a B.A. or B.Sc.

If you have the grade point average to be eligible for an honours programme (5.0), you will be listed as an honours student for administrative purposes. The Ordinary programme is not accredited by the CSAC.

Specialized Honours
Students selecting this programme take more courses in computer science and mathematics than for a major programme during their four years of study.

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 programme as students must maintain a Science grade point average of 6.0.

BSc Honours Double Major (formerly Combined Honours - prior to 99/00)
The intention of a combined programme is for students to major in two subjects while maintaining a 5.0 average. In general, students complete enough course work in each subject to obtain the equivalent of an honours degree. Degrees may require Honours Double Major students to take more than the minimum of 120 credits to satisfy the honours requirements of each subject.

BA Double Major/Major-Minor
In the Faculty of Arts a combined programme consists of either a double major or a major and a minor. In the latter case computer science can be either the major or the minor subject. Consult advisors in both departments if you are planning a combined programme.

BA Honours Double Major Programme in Computer Science and Mass Communications Studies
This double major programme differs from a standard double major programme in that the second major is in an interdisciplinary programme. In this double major programme, students are required to complete at least 7 Computer Science courses (i.e. 42 credits), two of which must be at the 4000 level. Students are also required to

complete 6 courses in Mass Communications Studies, one of which must be at the 4000 level.

BA Honours Double Major Programme in Computer Science and Women's Studies

The requirements of this programme are similar to those stated for the double major in mass communications studies except the second major is in women's studies.

Elective Courses

Students in Computer Science sometimes feel their study in this discipline is quite isolated from the other programmes 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 which 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 programmes, except the BA minor, now require at least 30 credits from non-COSC and non-MATH courses.

Industrial Internship Programme

The Industrial Internship Programme allows students to take a year off from their studies to gain valuable job experience and earn money while completing their degree. Students resume their studies when the internship term finishes. (A student's file becomes inactive during their internship; students must arrange to have their file reactivated in the spring of the year in which they will return.) Job assignments can be from 8 to 16 months depending on the corporation. Participating corporations include IBM, Celestica and Nortel for example.

Any 2nd or 3rd year student with a B average (or better) in MATH and COSC courses may apply to this programme. Qualifications may also depend on specific job postings. Job opportunities are posted in the fall, and students apply by submitting a completed application form, a nonrefundable fee, their resume and York University transcripts to the Computer Science Administrative Office (126 CCB) at the beginning of January. The Department forwards the student's application to companies which the student has selected. The companies select from these applications the students they wish to interview, and the Department then arranges on-campus interviews. Job offers are typically extended in the February/March time frame for positions starting in May.

In cases where the internship project involves significant learning of an academic nature students may receive credit for work done in connection with the internship by enrolling in the project course, COSC4080 3.0. The same rules apply for such internship projects as for the usual COSC4080 3.0 projects except that the work is not done at York University and is not done in a single academic term.

Students who are interested in doing a project as part of their internship should contact the course director of COSC4080 3.0 when they have enough experience on the job to be able to suggest a project topic which is compatible with the work they are asked to do and which has significant academic content.

It is the student's responsibility to ensure that the employer is willing to have the student report on her/his work in written and oral presentations. Work which cannot

be generally disclosed is not suitable for a COSC4080 3.0 project. Email intern@cs.yorku.ca

Admission to the Graduate Programme in Computer Science

Admission to the MSc programme is highly competitive.

The ideal preparation for graduate studies in Computer Science is the completion of the Specialized Honours Programme 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.

Need to upgrade a degree?
If you already have a Computer Science degree then you would upgrade, if necessary, your background to be equivalent to the Specialized Honours Programme in Computer Science. A comparison of the degree programme you completed with the Specialized Honours programme will show you what you are missing.

If you have an Ordinary 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 Programme Supplemental Calendar and the Graduate Programme 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 Specialised 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 Specialised Honours degree. If this number is greater than 3 go to the York Admissions Office (Atkinson College building, room 150) and apply for admission to

the undergraduate degree programme. If the number is 3 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 Programme Director or the Graduate Programme Director. Neither person can officially tell you how many courses in the undergraduate programme you will get credit for, and they cannot estimate it any better than you can yourself.

The Service Programme

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

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 work in the Ariel Lab, the Department of Computer Science undergraduate computing laboratories. First and second year students have access to 37 colour NCD X-terminals, and 20 Sunworkstations, including 15 Sun Ultra-1 workstations. Third and fourth year students are granted access to the Senior Lab consisting of 20 Sun Ultra-5 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 a CRS+ robot arm, an RWI B12 mobile robot, Sun workstations with video acquisition hardware, and NCD terminals.
- The Digital Logic Laboratory provides hands-on experience in computer design.
- The Real-Time Laboratory provides a high-performance Sun workstation, a 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 provides video and audio and multi-media author tools.

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, and course work. Students are automatically given access to the machines required to do their course work. However, students who would like to work on a project outside of assigned class work may ask a faculty member to act as their supervisor, and if necessary, a special account can be arranged for that project.

Computer Use Policy

Working in a laboratory situation requires cooperative behaviour which does not harm other students by making any part of the department's computer systems unusable such as locking out terminals, running processes which require lots of network traffic (such as playing games on multiple terminals), or using the facilities to work on tasks which 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 programme. 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).

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. Written information and programme 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 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 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 Programme Requirements

Whenever new programme 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 credits and the grade from such courses is not included in calculating the Computer Science grade-point-average.

COSC 1020 3.0
Introduction to Computer Science I (same as AS/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)].

Recommended: Previous programming experience; for example, a high school programming course or SC/AS/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/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.*

Advice: If it is possible, we suggest that you enrol for the summer or winter term.

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, but 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 SC/AS/COSC1020 3.0 and SC/AS/COSC1030 3.0. This course does not count as a Computer Science major credit.

Prerequisites: none

Degree Credit Exclusions: SC/AS/COSC1540 3.0. This course is not open to any student who has passed or is taking SC/AS/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.

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 Exclusions: COSC1530 3.0, SC/ACMS 1010 2.0.

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 for Science students).

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/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), trees
- Graphs: representations; transitive closure; graph traversals; spanning trees; minimum path; flow problems

Prerequisites: general prerequisites.

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

This course intends to provide students basic understanding of computers at the lowest levels of this structure. The ways in which data are represented in memory and transformed by machine instructions are explored.

The major functional blocks of a computer including main memory, control unit, ALU, input/output bus structures, interrupt system, DMA channels, and peripheral devices are studied. Some assembler programming and microprogramming will be required.

Suggested Reading:
- Tanenbaum, A.S., *Structured Computer Organization,* 3rd ed., Prentice-Hall, 1990.
- Stallings, Wm., *Computer Organization and Architecture,* 2nd ed., Macmillan, 1990.

Prerequisites: general prerequisites.

COSC 2031 3.0
Fundamentals of Unix, C and C++

1 The programming languages C and C++ are taught as a second programming language. The course assumes some programming experience, typically that

obtained in the first-year computer science course. The emphasis is on practical exploration of a particular strength of C, rather than its use as a general-purpose language. C is a tool-making tool for working at a low-level; it is a programming language designed for, and used in, the writing of utilities that access the primitives of the underlying machine architecture and the operating system.

2 The UNIX programming environment is considered at both

a. the command level (man 1): shells, utilities, filters, etc., and

b. the system call (man 2) and standard library (man 3) levels.

The focus in [2b] is on some Unix operating system user-interface primitives, e.g. fork/exec, file descriptors, read/write, etc. The emphasis here, as in [1], I s on practical exploration.

3 C++ is first introduced as an improved C, e.g. with its use of reference parameters. Its object-oriented features are then looked at in comparison to C and to (object-oriented) Java, which students should know from introductory courses.

Suggested Readings:

- Kernighan and Ritchie, The C Programming Language.
- Richard Stevens, Advanced Programming in the UNIX Environment.
- Ira Pohl, C++ for C Programmers.

Prerequisites: general prerequisites.

Course Descriptions: 3000-Level

General Prerequisites

Before enrolment is permitted in any 3000-level computer science course (except service courses – 35xx x.x) the following prerequisites must be met.

- COSC2011 3.0 completed.
- One of COSC2001 3.0 or COSC2021 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 for Science students).
- MATH1300 3.0 and MATH1310 3.0 completed.

- One of MATH2090 3.0, MATH2221 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: g*eneral prerequisites, including MATH2320 3.0 (SCS students may enrol without MATH2320 3.0 or concurrently with MATH2320 3.0)

NOTE: This course is required of all specialized honours students in computer science – except those in the SCS stream.

COSC 3111 3.0
Introduction to Program Verification

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

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

Topics covered will include the following:

- The role of formal verification in the software life-cycle; verification vs. testing and validation.
- Introduction to propositional calculus; checking for tautologies and contradictions; annotating code with assertions.
- Symbolic execution; proving relative correctness for small code segments; establishing termination.
- Creating specifications with quantifiers; translating specifications into code.

Suggested 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

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 Values 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: for Computer Science majors - general prerequisites, including MATH2221 3.0;
for others - COSC1540 3.0 or COSC2011 3.0 or COSC2031 3.0;
MATH1010 3.0 or MATH1014 3.0 or MATH1310 3.0;
MATH1025 3.0 or MATH2021 3.0 or MATH2221 3.0.

Degree Credit Exclusion: 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 3211 3.0
Data Communication

- Physical and Electrical. How complex periodic signals propagate through guided media such as twisted pairs, co-axial cable, point-to-point microwave links, and fibre optics, and through the unguided medium of broadcast electromagnetic radiation. Frequency-domain (Fourier, spectral) analysis. Noise. Limiting relationship between signal-rate and bandwidth (Nyquist) and between data-rate and bandwidth and signal-to-noise ratio (Shannon).
- Data encoding. How analogue and digital data are carried in some encoded form by analogue and digital signals.

Data	Signal
Analogue	Analogue : amplitude, frequency, phase modulation.
Digital	Analogue : amplitude, frequency, phase shift keying.

Analogue	Digital	:	sampling, quantisation, pulse code modulation.
Digital	Digital	:	NRZ-L, NRZI, Bipolar-AMI and Manchester.

- Data Link. How frames of data bits are transmitted in a controlled and reliable way between physically (i.e. directly) linked nodes in a network. Protocols for flow control, and error detection and control.
- Multiplexing and switching. How unrelated data streams may share common pathways. Time-division and frequency-division multiplexing. Circuit switching. Space-division switching. Packet switching.
- Networks. LAN (local area networks). Ethernet (CSMA/CD), token bus and token ring. Virtual circuits and datagrams.
- Basics of the TCP/IP protocol suite.

Prerequisites: general prerequisites, including COSC2021 3.0 and MATH2090 3.0

COSC 3212 3.0
Computer Networks
This course covers the upper layers in the OSI(TCP/IP) reference models. Topics covered include:

- Introduction: Local area networks, high speed local area networks, Metropolitan Area networks, and wireless networks.
- Network layer: Routing, congestion control, traffic shaping, Internetworking, IP, IPv6, and network layer in ATM networks.
- Transport layer: Transport protocols, transport connection management, TCP, UDP, and the ATM Adaptation Layer (AAL) protocols.
- Application layer: Remote Procedure Calling (RPC), network security, Abstract Syntax Notation (ASN), multimedia and data compression.

Prerequisites: general prerequisites; COSC3211 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 meet the needs of the student who wishes to learn to program in a 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 program which does not work is undoubtedly wrong; but a program which does work is not necessarily right. It may still be wrong because it is hard to understand or because it is hard to maintain as the program requirements change; or because its structure is different from the structure of the problem." (M. A. Jackson)

This course introduces the topic of software design through lectures, supplementary readings and a set of small design problems. The course deals with the problem of designing software that can be used, understood and modified by people other than the original designer.

Software design is in itself a large topic as design can deal with various classes of programs and systems: small, medium and large; batch; real time; distributed; and interactive (visual and graphical). Every design class has its own problems. In this course we deal with small to medium programs and small systems that work without critical time constraints (although time will be considered).

We examine design methods such as JSP (Jackson System Programming), Data Flow, SADT (Structured Analysis and Design Technique), top down, bottom up and structured design methods. We show how theoretical notions from finite state machines and grammars are related to and used in the design process.

Some of the low level techniques we look at are: abstract data types; backtracking; divide and conquer; structure clash resolution; process inversion; coroutines; and error handling.

Design issues are related to the other phases of developing a program: requirements analysis, specification, implementation, testing, and maintenance.

Upon leaving the course, you can expect to be able to design, implement and modify programs and systems of programs which transform sequences from one form to another. You will understand the role of tools and frames in the design process. You will be able to evaluate program designs and design methods.

Prerequisites: general prerequisites, including COSC 2001 3.0 and MATH2090 3.0; COSC2031 3.0

COSC 3321 3.0
Operating System Fundamentals
This course is intended to teach students the fundamental concepts that underlie operating systems, including multiprogramming, concurrent processes, CPU scheduling, deadlocks, memory management, file systems, protection and security. Many examples from real systems are given to illustrate the application of particular concepts. At the end of this course, a student will be able to understand the principles and techniques required for understanding and designing operating systems.

Prerequisites: general prerequisites, including COSC2021 3.0; COSC 2031 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 and practice of Object-Oriented software construction, one of the major areas of software engineering practice and research.

A source program in an object-oriented programming environment is based on independently defined abstract program units called object classes or types; the object instances of the classes are created in the program and interact through message passing at run-time. Each object is a program unit which packages both a coherent collection of data elements and the methods (routines) for their manipulation. The object encapsulates its data elements, which means that the object class has an interface which defines all and only what is visible about the object instances to other objects in the program environment. These techniques promote the reuse of reliable code and reduce the frequency of the kinds of programming which arise in attempting to integrate the different parts of a complex program.

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, Delphi.

- Abstract Data Types as a basis for information hiding. Other software engineering principles such as the open-closed principle and the single point principle.
- The construction of a hierarchy of data types in which types are derived from one or more ancestor types by inheritance and refinement.
- A discussion of overloading, redefining, hiding of member functions.
- Design by contract including preconditions, postconditions class invariants, loop invariants, etc. Contracts and exception handling. Contracts and testing.
- Polymorphism and genericity as an approach to code reuse.
- Static versus dynamic binding of methods to method calls.
- Techniques of object-oriented design: identifying objects and classes; identifying semantics of objects and classes; identifying class hierarchies; identifying when to use polymorphism.
- The BON notation for seamless OO design and analysis.
- Multiple inheritance and repeated inheritance.
- Object persistence. Class STORABLE.
- Class library design to provide reusable software components.

Prerequisites: general prerequisites

Degree Credit Exclusion: COSC3010A 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 which 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; MATH2320 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/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/ITEC3461 3.0)
Human-Computer Interaction
This course introduces the concepts and technology necessary to design, manage and implement interactive software. Students work in small groups and learn how to design user interfaces, how to realize them and how to evaluate the end result.

Introduction (goals, motivation, human diversity)
Theory of Human-Computer Interaction (golden rules, basic principles, guidelines)
The Design Process (methodologies, scenario development)

Expert Reviews, Usability Testing, Surveys and Assessments
Software Tools (specification methods, interface-building tools)
HCI Techniques
- Interaction Devices (keyboards, pointing devices, speech recognition, displays, virtual reality devices...)
- Windows, Menus, Forms and Dialog boxes
- Command and Natural Languages (command line and natural language interfaces)
- Direct Manipulation and Virtual Environments
- Manuals, Help Systems, Tutorials
- Hypermedia and the World Wide Web (design, creation, maintenance of Documents)

Human Factors
- Response Time and Display Rate
- Presentation Styles: Balancing Function and Fashion (layout, colour)
- Societal Impact of User Interfaces (information overload)

Computer Supported Cooperative Work (CSCW, synchronous and asynchronous)
Information Search and Visualization (queries, visualization, data mining)

Prerequisites: ITEC/COSC2011 3.0

Degree Credit Exclusion: Not open to students who successfully completed AS/SC/COSC4341 3.0, AS/SC/COSC4361 3.0 before FW99

COSC 3530 3.0 (x-listed BC3030 3.0)
Technical and Professional Writing
This writing-intensive course is for upper-year students enrolled in Computer Science, Space and Communication Sciences, and other Science-related fields. Students will develop confidence and competence in professional and technical writing. Focus is on communication of complex information in a clear, sensible style. Articles, end-user documentation and technical reports will be critically evaluated in class. Outside speakers from industry will be invited to provide a real-world perspective on needs.

Note: This course counts as elective Science credits toward satisfying Faculty degree requirements but does not count as Computer Science major credit. The grade for this course is not included in the calculation of the Computer Science grade point

average (GPA). However, this course is recommended by the Computer Science department for students who wish to pursue a professional career.

This course may be taken using the Pass/Fail grading option. Science students who wish to take it on a Pass/Fail basis must have completed at least 24 credits and have taken no more than 9 previous Pass/Fail credits.
Web site: www.yorku.ca/bethune/courses/3030.html

Corequisite: concurrent enrolment in at least one course in the third or fourth year of Computer Science, Physics, other applied science or permission of the instructor.
English language proficiency is expected.

Degree Credit Exclusions: SC/BC3050 3.0.

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 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 for Science students).
- 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 programme, 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: COSC 4001 6.0

COSC 4101 3.0 (x-listed COSC 5101 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 COSC 3101 3.0; MATH2320 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; MATH2320 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 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 and COSC3408 3.0

COSC 4242 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:

EATS4020 3.0, MATH4130B 3.0, MATH4830 3.0, PHYS4060 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)
Language Processors
This course is concerned with a variety of theoretical and practical questions that are raised by the need to implement programming languages. The implementation of a programming language is accomplished by writing a computer program that functions as a "translator" or "language processor". Language processors can be categorized as compilers, cross-compilers, interpreters, assemblers, and pre-processors. However, the different kinds of translators generally exhibit a common internal

structure consisting of standardized "phases": the scanner, parser, and semantics. Because of this common structure, it is appropriate to study language processors as a unified group. However, the course emphasizes the most important class of language processors: compilers.

As a practical exercise, students are required to design and implement components of a compiler.

Topics covered may include the following:

- Phases of translation, single-pass and multi-pass translation, phase interleaving.
- Lexical analysis, or scanning: finite-state automata, practical simplifications of finite-state automata.
- Syntactic analysis, or parsing: shift-reduce parsing, operator precedence parsing, top-down parsing, bottom-up parsing, backtracking, look-ahead, efficiency considerations.
- Context conditions, semantic analysis, code generation.
- Data structures for translators, symbol tables.
- Optimization: local optimizations, register optimization, global optimization, control flow analysis, data flow analysis.
- Error analysis: error trapping and recovery.

Prerequisites: general prerequisites, including COSC 3301 3.0; COSC2031 3.0

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, and COSC3111 3.0 or COSC3321 3.0.

COSC 4321 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 3321 3.0.

COSC 4331 3.0 (x-listed COSC 5331 3.0)
Computer Graphics
This course introduces the student to the fundamental algorithms of 3-D computer graphics and image synthesis.

The first half of the course covers window systems, display hardware, graphical primitives, scan conversion, two and three-dimensional transformations and the mathematics of planar geometric projection.

The second half concentrates on raster algorithms and image synthesis. Some of the topics include visible surface algorithms, modeling, shading, global illumination, anti-aliasing and texture mapping.

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: MATH2221 3.0

COSC 4341 3.0
Interactive System Design
Recent studies have shown that on average, 50% of the costs of modern application programs come from the development of the program's user interface. At the same time, users of programs are becoming increasingly aware and critical of poorly designed user interfaces. The modern system designer must, therefore, be capable of creating systems that are easy to use and that truly address the needs of the users, while still managing to remain within budget.

This course introduces the concepts and technology necessary to manage the design and implementation of interactive software. Students will learn methods for designing software matched to the goals, tasks and skills of the eventual users of the system. A

development methodology is presented in which each development stage involves evaluation of the quality of the system design. Units in the course will cover User Needs Analysis, User Interface Design, Task-Oriented Specification, Architecture Design of Interactive Systems, System Implementation, and User Testing and Evaluation.

Prerequisites: general prerequisites.

Degree Credit Exclusion: SC/AS/COSC3431 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 modelling, simulation, specification, analysis, design and verification of such real-time programs. The objective of the course is to expose the student to current techniques for formally proving the correctness of real-time behaviour of systems.

Topics covered may include the following:

- Techniques for expressing syntax and semantics of real-time programming languages
- Modelling real-time systems with discrete event calculi (e.g. Petri net and state machine formalisms)
- Specification of concurrency, deadlock, mutual exclusion, delays and timeouts
- Scheduling of tasks to meet hard time bounds.

- CASE tools for analysis and design. At the end of the course the student will be able to model and specify real-time systems, design and verify correctness of some real-time systems.

Prerequisites: general prerequisites, including COSC 3311 3.0 or COSC3321 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, and examples from real-time programming languages (Ada and Java).

Prerequisites: general prerequisites, including COSC 3301 3.0 or COSC3311 3.0 or COSC3321 3.0

COSC 4361 3.0
Human-Computer Communication:
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.

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 which 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 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, including COSC3412 3.0 or 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; MATH2221 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

Required Mathematics Courses

The introductory courses MATH1090 3.0, MATH1300 3.0, and MATH1310 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 programme - MATH2090 3.0, MATH2221 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
MATH1090 3.0	MATH1120 3.0
MATH1300 3.0	MATH1000 3.0, MATH1013 3.0
MATH1310 3.0	MATH1010 3.0, MATH1014 3.0
MATH2221 3.0	MATH1025 3.0, MATH2021 3.0

MATH 1090 3.0
Introduction to Sets and Logic (formerly MATH1120 3.0)
The syntax and semantics of propositional and predicate logic. Applications to program specification and verification. Optional topics include set theory and induction using the formal logical language of the first part of the course.

By taking this course, students will master the syntax and manipulations of propositional and predicate logic, as well their informal semantics. The proper understanding of propositional logic is fundamental to the most basic levels of computer programming, while the ability to correctly use variables, scope and quantifiers is crucial in the use of loops, subroutines, and modules, and in software design. Logic is used in many diverse areas of computer science including digital design, program verification, databases, artificial intelligence, algorithm analysis, and software specification. We will not follow a classical treatment of logic. Instead we will use an “equational” treatment. This equational approach will also be the basis for the topics in discrete mathematics treated in MATH2090.

Prerequisite: One OAC in mathematics or equivalent or AK/MATH1710 6.0.

Degree Credit Exclusions : AS/SC/MATH1120 3.0, AS/SC/AK/MATH1190 3.0.
This course is not open to any student who has taken or is taking any 3000-level or higher-level Math course.

MATH 1300 3.0
Differential Calculus With Applications
Limits, derivatives with applications, antiderivatives, fundamental theorem of calculus, beginnings of integral calculus.

Other topics include continuity, the Mean Value Theorem, curve sketching, L'Hospital's rule, maxima and minima, and (time permitting) applications of integration theory.

The final grade may be based on assignments, quizzes, class tests and a final examination worth at least 30%.

Prerequisites: OAC Calculus or AS/SC/MATH1500 3.0 or

AS/SC/MATH1515 3.0 or
AK/MATH1710 6.0 or equivalent.

Degree Credit Exclusions: AS/SC/MATH1000 3.0, AS/SC/MATH1013 3.0, AS/SC/MATH1505 6.0, AS/MATH1530 3.0, AS/AK/MATH1550 6.0, AS/ECON1530 3.0, AK/MATH1410 6.0,

MATH 1310 3.0
Integral Calculus with Applications
Transcendental functions, differential equations, techniques of integration, improper integrals, infinite series. Offered in both terms.

This is the second in a series of introductory calculus courses. It is designed to follow MATH 1300 3.0.

Other topics include infinite sequences. Differential equations and their applications will only be discussed to the extent that time allows.

The final grade may be based on assignments, quizzes, class tests, and a final examination worth at least 30%.

Prerequisites: One of AS/SC/MATH 1000 3.0, AS/SC/MATH 1013 3.0, AS/SC/AK/MATH 1300 3.0, or, for non-Science students only, six credits: AS/MATH 1530 3.0 and AS/MATH 1540 3.0, or AS/AK/MATH 1550 6.0; or AS/ECON 1530 3.0 and AS/ECON 1540 3.0.

Degree Credit Exclusions: AS/SC/MATH 1010 3.0, AS/SC/MATH 1014 3.0, AS/SC/MATH 1505 6.0, AK/MATH1410 6.0.

MATH 2090 3.0
Introduction to Mathematical Logic
(1) The title of this course no longer fits its content, and is likely to be changed after we go to press.
(2) The official York Calendar description, and list of prerequisites, are also superannuated, and fail to appear here because they are being updated.
(3) This course has Introduction to Logic for Computer Science (MATH1090, first offered in Fall of 1998) as a strict prerequisite. Currently this strictness is unofficial, but it is expected to receive faculty approval in the near future. Students who lack this prerequisite must check with the department before enrolling.

A partial indication of the relevance of formal logic to programming is given in the course entry for COSC 3111 in this year's supplemental calendar issued by the Department of Computer Science.
"Every program implicitly asserts a theorem to the effect that the program will do what its documentation says it will". Proving that theorem "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".

MATH 2090 is a continuation of MATH 1090 (Introduction to Logic for Computer Science), and will use the mathematical logic learned in that course to study selected topics in discrete mathematics. Students wanting further exposure to discrete math may consider MATH 2320 3.0.

Topics will include sets, relations and functions, induction, and a study of the integers.

MATH 2221 3.0
Linear Algebra with Applications I
Systems of linear equations, linear and affine subspaces of Euclidean, the Gauss--Jordan algorithm, matrices and matrix algebra, determinants, vector space concepts for Euclidean n-space (linear dependence and independence, basis, dimension, etc.), various applications.

Linear algebra is a branch of mathematics which is particularly useful in other fields and in other branches of mathematics. Its frequent application in the engineering and physical sciences rivals that of calculus. Computer scientists and economists have long recognized its relevance to their discipline. Moreover, linear algebra is fundamental in the rapidly increasing quantification that is taking place in the management and social sciences.

Finally, ideas of linear algebra are essential to the development of algebra, analysis, probability and statistics, and geometry.

This course and MATH 2222 3.0 together provide a standard full-year introduction to linear algebra. While our focus will not be excessively theoretical, students will be introduced to proofs and expected to develop skills in understanding and applying concepts as well as results. Applications will be left mainly for MATH 2222 3.0.

Note that MATH 1540 3.0 may not be taken for credit by anyone who is taking, or anyone who has taken, MATH 2221.

Prerequisite: OAC algebra or any university mathematics course.

Degree Credit Exclusions: AS/SC/MATH 1025 3.0, AS/SC/MATH 2021 3.0, AK/MATH 2220 6.0

MATH 2320 3.0
Discrete Mathematical Structures
This course covers the algebraic and combinatorial structures that are needed in computer science. Topics include set theory, functions, relations, combinatorics, elements of graph theory, posets, lattices, Boolean algebras, monoids, groups, morphisms, congruence relations. Intended primarily, but not exclusively, for students in Computer Science.

Consultation with the Department of Computer Science has led to the following list of topics for emphasis: "Big oh" notation, complexity of formulae and algorithms, modular arithmetic, recursive definitions, general inductions, counting principles, recurrence relations and methods for solving them, trees and simple graph theory. The emphasis will include examples arising from algorithms and the ability to carry out analysis, problem solving, proofs and calculations which will be required in upper level courses.

The course does not require previous knowledge of computer science. A student of mathematics should enjoy this introduction to a variety of mathematical topics, many of which are not covered elsewhere.

This course is a prerequisite for COSC 3101 3.0, COSC 3402 3.0, COSC 4101 3.0, COSC 4111 3.0.

Prerequisite: AS/SC/AK/MATH 1090 3.0 or any 2000-level MATH course (without second digit 5) or permission of the course director.

Degree Credit Exclusions: AK/MATH 2440 6.0, AK/MATH2442 3.0.

MATH 2560 3.0
Elementary Statistics I
Displaying and describing distributions, normal distribution. Relationships between variables, regression and correlation. The need for design, experimental design and sampling design. Sampling distributions, bias, variability. Probability models, random variables, probability laws.

Statistics is a collection of methods for observing and analyzing numerical data in order to make sensible decisions about them. In these courses the basic ideas of the analysis of data and of statistical inference will be introduced.

Little mathematical background is required; high school algebra is sufficient. Mathematical proofs will be minimal; reasoning and explanations will be based mostly on intuition, verbal arguments, figures, or numerical examples. Most of the examples will be taken from our daily life; many deal with the behavioural sciences, while others come from business, the life sciences, the physical sciences, and engineering.

Although students will be making some use of the computer to calculate statistics, to create statistical plots, and to obtain a better appreciation of statistical concepts, (no previous experience in computing is required). Students will receive in class all the necessary instruction about how to use the statistical computer package Minitab.

Students who have taken MATH 2560 3.0 will normally take MATH 2570 3.0 in the second semester, where they will continue to investigate many basic statistical methods.

Prerequisite: Ontario Grade 12 Advanced Mathematics.

Degree Credit Exclusions: AS/SC/MATH 1131 3.0, SC/BIOL 3080 3.0, SC/BIOL 3090 3.0, AS/ECON 2500 3.0, AS/SC/GEOG 2420 3.0, AS/SC/KINE2050 3.0, AS/SC/PHED 2050 3.0, AS/SC/PSYC 2020 6.0, AS/SC/PSYC 2021 3.0AS/SOCI 3030 6.0, AK/MATH 1720 6.0, AK/MATH 2430 6.0, AK/BIOL3080 6.0, AK/BIOL3080 3.0, AK/PSYC2510 3.0.

Not open to any student who has successfully completed AS/SC/MATH 2030 6.0.

Degree Requirements - Faculty of Arts

Students should consult the York University Undergraduate Calendar for full details of the degree requirements and relevant regulations (check the Calendars link at www.yorku.ca).

Course Types

Degree requirements in the Faculty of Arts refer to the following categories of courses:

I. **General Education Courses** - Intended to provide a broad interdisciplinary perspective, "General Education" courses, are offered by three academic units, the Divisions of Humanities, Natural Science, and Social Science. The Faculty of Arts Foundations courses are part of the General Education requirement and are affiliated with the appropriate Faculty of Arts Colleges. These courses emphasize critical skills, especially writing, and provide a supportive learning environment. The connection to one of the Faculty of Arts Colleges - Calumet, Founders, McLaughlin, Stong, or Vanier - enables students to participate more fully in college life and co-curricular events.

II. **Major (and Minor) Courses** - In addition to taking courses which contribute to their broad knowledge, students are required to specialize in a specific subject or combination of subjects. The area of primary concentration is known as the "Major"; an area of secondary concentration (if any) is known as the "Minor". It is possible to have two "Majors".

III. **Elective Courses** - intended to broaden the educational experience of students beyond their area of specialization, electives include most courses which a student does not use to fulfil either General Education requirements or Major/Minor requirements. Courses not considered to be electives are:
   a) Major courses taken above the required number;
   b) non-Major courses taken within the Major subject (e.g. most Atkinson COSC courses);
   c) courses outside the Major taken to fulfil major requirements (e.g., AS/MATH1090 3.0 and other MATH courses required for Computer Science);

d) courses which are cross-listed or designated as equivalents or exclusions for courses offered by the Major Programme.

IV. **Upper-Level Courses** - courses at the 3000-level and/or 4000-level. Honours degrees require at least 18 credits at the 4000-level and at least 36 credits at the 3000- and 4000-level.

Programme Types

The Faculty provides for the following types of programmes in Computer Science:

Honours Programmes - are 120 credit programmes, which require more specialization, a higher minimum performance, and in some cases, different courses than does an Ordinary programme.

In order to graduate with an Honours Degree, students must successfully complete a minimum of 120 credits which fulfil one of the following requirements [a), b), c), or d)]:

a) Specialized Honours:

I. General Education: See General Education Requirements below;

II. Major Requirements:
- COSC1020 3.0, COSC1030 3.0, COSC2001 3.0, COSC2011 3.0, COSC2021 3.0,
- COSC3101 3.0 and 18 more credits at the 3000-level satisfying breadth in COSC (see below),
- COSC4101 3.0 or COSC4111 3.0, and 9 more credits at the 4000-level in COSC,
- 6 more credits at the upper level (3000- or 4000-) in COSC,
- MATH1090 3.0, MATH1300 3.0, MATH1310 3.0, MATH2090 3.0, MATH2221 3.0 and MATH2320 3.0,
- and at least 30 credits that are neither COSC or MATH;

III. Elective Courses: at least 18 credits;

IV. Upper-Level Courses:

a) 3000-level and 4000-level courses: at least 36 credits at the 3000-level or 4000-level;
b) 4000-level courses: at least 18 credits of these upper-level courses [including at least 12 credits in the Major] must be at the 4000-level;

V. In-Faculty Courses: See In-Faculty Courses in the York University Undergraduate Calendar;

VI. Standing Requirements: In addition to fulfilling the requirements described above, students in this programme must achieve satisfactory academic standing to enter, proceed, and graduate.

b) Honours (Major):

I. General Education: See General Education Requirements below;

II. Major Courses:
- COSC1020 3.0, COSC1030 3.0, COSC2001 3.0, COSC2011 3.0, COSC2021 3.0,
- 15 credits at the 3000-level satisfying breadth in COSC and 12 credits at the 4000 level,
- MATH1090 3.0, MATH1300 3.0, MATH1310 3.0, MATH2090 3.0, and one of MATH2221 3.0 or MATH2320 3.0,
- and at least 30 credits that are neither COSC or MATH;

III. Elective Courses: at least 18 credits;

IV. Upper-Level Courses:
a) 3000-level and 4000-level courses: at least 36 credits at the 3000-level or 4000-level
b) 4000-level courses: at least 18 credits of these upper-level courses [including at least 12 credits in the Major] must be at the 4000-level;

V. In-Faculty Courses: See In-Faculty Courses in the York University Undergraduate Calendar;

VI. Standing Requirements: In addition to fulfilling the requirements described above, students in this programme must achieve satisfactory academic standing to enter, proceed, and graduate.

c) Honours (Minor):

A Minor in Computer Science must be combined with a Major in a different subject. The Minor in Computer Science has the requirements listed under (2)(b).

I. General Education: See General Education Requirements below;

II. Major/Minor Courses: as defined by the specific programmes;
   a) Major: usually at least 42 credits in the Major, at least 12 credits of which must be at the 4000-level; and
   b) Minor (in COSC):
      - COSC1020 3.0, COSC1030 3.0, COSC2001 3.0, COSC2011 3.0, COSC2021 3.0,
      - 15 credits at the 3000-level satisfying breadth in COSC and 6 credits at the 4000 level,
      - MATH1090 3.0, MATH1300 3.0, MATH1310 3.0, MATH2090 3.0, and one of MATH2221 3.0 or MATH2320 3.0;

III. Elective Courses: Students who graduate in an Honours (Major/Minor) programme are deemed to fulfil the Elective Course requirement;

IV. Upper-Level Courses:
   a) 3000-level and 4000-level courses: at least 36 credits at the 3000-level or 4000-level
   b) 4000-level courses: at least 18 credits of these upper-level courses [usually including 12 credits - in the Major and 6 credits - in the Minor] must be at the 4000-level;

V. In-Faculty Courses: See In-Faculty Courses in the York University Undergraduate Calendar;

VI. Standing Requirements: In addition to fulfilling the requirements described above, students in this programme must achieve satisfactory academic standing to enter, proceed, and graduate.

d) Honours (Double Major):

Students may combine a Major in COSC (as defined above) with a Major in a different subject in the Faculty of Arts (either linked or unlinked), with an Honours Major in

some Fine Arts subjects or in Environmental Studies. Elective courses are deemed fulfilled in these programmes; there must be at least 30 credits that are not COSC and not MATH.

Ordinary Programme - in order to graduate with an Ordinary Degree a student must successfully complete a 90 credit programme as follows:

1. General Education: see General Education Requirements below;
2. Major courses: students must complete at least 33 credits in Computer Science, including:
   - AS/COSC 1020 3.0, AS/COSC 1030 3.0, AS/COSC 2001 3.0, AS/COSC 2011 3.0 and AS/COSC 2021 3.0,
   - 18 credits at the 3000-level satisfying the departmental breadth requirement (see note below),
   - In addition, students must complete AS/MATH 1090 3.0 and AS/MATH 1300 3.0 and AS/MATH 1310 3.0 and AS/MATH 2320 3.0;
3. Elective courses: at least 18 credits;
4. Upper-Level courses: at least 18 credits at the 3000-level or 4000-level and at least 12 credits of which must be in the major subject;
5. Standing Requirements: in addition to fulfilling the requirements described above, students in this programme must achieve satisfactory academic standing to enter, proceed, and graduate.

General Education Requirements

Students will select their General Education requirements in accordance with the following:

1. One 1000-level 9 credit Foundations course, in either the Division of Humanities or the Division of Social Science - to be taken within the first 24 credits;

2. One 1000-level 6 credit course in the Division of Natural Science - to be taken prior to graduation, and preferably within the first 42 credits;
3. One 2000-level 9 credit Foundations course, in either the Division of Humanities or the Division of Social Science. If the 1000-level 9 credit Foundations course is taken in the Division of Humanities, then the 2000-level 9 credit Foundations course must be taken in the Division of Social Science (and vice versa) - to be taken within the first 48 credits;
4. Breadth requirement - will be satisfied by successfully completing the General Education/Foundations courses described above. (Breadth requirement is described below.)

The Breadth requirement must be successfully completed before graduation and requires at least 6 credits from each of the following areas:

Area I	**Area II**
Classical Studies (Greek or Latin)	Anthropology
English	Economics
French Studies	Geography
History	Political Science
Humanities	Psychology
Languages, Literatures, & Linguistics	Social Science
Philosophy	Sociology

Note: Mathematics, Computer Science, and Kinesiology and Health Science courses, will satisfy neither requirement, unless they are cross-listed with a unit listed above.

Passed Courses:

A student who has received a passing grade for a course may not repeat that course or take an equivalent or excluded course for degree credit.

Failing Grades:

Where a student retakes a course in which he/she has previously received a grade of E or F and passes it, the passing grade is also calculated into the Grade Point Average.

Honours Standing In Degree Programmes

Qualifying for Honours -

a) Students With No Previous Post-Secondary Education: Students who enter the Faculty of Arts with no prior experience at a post-secondary educational institution (such as a university or college) are automatically enrolled in an Honours programme.

b) Transfer Students: Students who enter with prior experience at a post-secondary educational institution are enrolled in an Honours programme if their prior Cumulative Grade Point Average (including failed courses) is at least the equivalent of 5.0 on the York scale. (Note: Courses taken at other post-secondary institutions are not calculated as part of the student's grade point average at York, nor do they appear on the York transcript.)

Continuing in Honours - To continue in an Honours programme, students must maintain a Cumulative Grade Point Average of at least 5.0.

Graduating in Honours - To graduate in an Honours programme, students must pass at least 120 credits which meet Faculty of Arts and programme requirements. The Cumulative Grade Point Average for all courses taken must be at least 5.0.

Degree Requirements - Faculty of Pure and Applied Science

General Education Requirements

General education courses are required within all BSc programmes. These non-science courses provide a broad perspective on current scholarship and the diversity of human experience. The courses are also expected to enhance students' critical skills in reading, writing, and thinking and contribute to their preparation for post-university life.

All BSc candidates must complete a minimum of 12 credits from two different areas of study, including at least 3 credits from each area, subject to the restrictions noted below. For the purposes of this regulation, "different area" means offered by different academic units such as divisions, departments or Faculties and excluding courses offered by similar departments in different Faculties (such as English in the Faculty of Arts and Atkinson College). Subject to the restrictions listed below, courses in the following areas may be taken in the Faculty of Arts, Atkinson College or Glendon College.

Anthropology	Classical Studies *
Economics	English
History	French Studies *
Humanities	Geography **
Philosophy	Languages, Literature & Linguistics *
Political Science	Social Science
Sociology	

The following courses offered by the Faculty of Environmental Studies and the Faculty of Fine Arts may also be taken to satisfy Faculty of Pure and Applied Science general education requirements: ES/ENVS1000 6.0, FA/INFA1900 6.0, FA/INFA2900 6.0 and FA/VISA2550 6.0.

General education courses are normally taken at the 1000 or 2000 level, but higher-level courses are acceptable, subject only to prerequisites and course access specifications for enrolment.

Permission may be granted by the Office of Science Academic Services, on an individual basis, for a student to take a course outside the areas and Faculties listed above for general education credit, subject to the course fulfilling the Faculty of Pure and Applied Science breadth and critical skills requirements for general education courses, the student having the appropriate prerequisites and the course access specifications permitting enrolment. A student who is in doubt regarding whether or not any specific course will fulfill the Faculty of Pure and Applied Science general education requirements should consult the Office of Science Academic Services.

Restrictions

1. Courses whose major focus is increased facility in the use of a language cannot count as general education courses. Such courses are offered in the departments marked with an * above.

2. Quantitative courses focusing on techniques of mathematics or statistics cannot count as general education courses. For example, this applies to some Economics courses.

3. Certain other types of courses cannot be used to satisfy general education requirements. In particular,

   (i) courses which are cross-listed as SC courses or which are eligible for SC credit cannot count as general education courses; and

   (ii) Geography courses (**) cannot be used to satisfy general education requirements for BSc candidates majoring in Geography.

Note: General education courses may not be taken on a pass/fail basis (see “Pass/Fail Grading Option" in Science section III of the University Undergraduate Calendar).

General Regulations

1. All students are required to observe the regulations of the University. Unless otherwise stated, any changes in regulations become effective as announced. This policy is not meant to disadvantage students as they proceed through their studies, including those who have completed a number of courses. It is intended to ensure that their preparation for courses is appropriate and current. Students should consult closely with departments and the Faculty through the advising process. (Students admitted to the Faculty prior to September 1995 are strongly advised to follow current regulations, but may elect to graduate under the regulations of the year in which they were admitted to the Faculty.)

2. It is the student's responsibility to enrol in only those courses for which the student has successfully completed all designated prerequisites and to take concurrently all specified corequisites not already completed successfully.

3. All degree candidates are required to indicate a choice of degree programme (Ordinary, Specialized Honours, or Combined Honours) upon successful completion of 24 credits. A minimum cumulative credit-weighted grade-point

average of 5.0 over all Science courses completed is required for Honours programmes. For information about changing degree programmes see the University Undergraduate Calendar.

4. Students admitted to York University for 1999/2000 and subsequent years, the Senate of York University will require a minimum overall grade-point average of 4.0 in order to be eligible to graduate in an undergraduate Ordinary degree programme.

5. Students admitted to York University for 1999/2000 and subsequent years, the Senate of York University will require a minimum overall grade-point average of 5.0 in order to be eligible to graduate in an undergraduate Honours degree programme.

6. All BSc degree candidates in Honours and Ordinary programmes must successfully complete the following minimum requirements, normally at the 1000 level:

   at least 24 Science credits, excluding SC/CHEM1500 4.0, SC/CHEM1520 4.0, SC/MATH1500 3.0, SC/MATH1510 6.0, SC/MATH1515 3.0, SC/MATH1525 3.0, SC/PHYS1510 4.0, and all Natural Science courses, and including at least 2 credits in introductory computer science, 6 credits in approved mathematics courses, and 12 credits in courses with laboratories;

   12 general education credits (see "General Education Requirements" in this section of the Calendar).

7. Ordinary Programme - all B.Sc. degree candidates in all Ordinary programmes must, through registration in courses at York University or elsewhere deemed creditable towards the B.Sc. degree,

   a) satisfy regulations 2, 3, 4 and 6;

   b) present a total of at least 90 passed credits of which

      · a minimum of 66 must be earned in Science courses,

      · a minimum of 24 must be earned in one major Science subject area (except in the case of General Science),

- a minimum of 18 must be earned in courses at the 3000-level or higher;

c) satisfy the programme of study requirements specified below for the Ordinary Programme.

8. To declare Honours requires successful completion of at least 24 credits and a minimum cumulative credit-weighted grade-point-average of 5.0* over all Science (SC) courses completed.

To proceed in each year of an Honours BSc programme requires a minimum cumulative credit-weighted grade-point average of 5.0* over all Science (SC) courses completed.

To graduate in an Honours BSc programme requires successful completion of all Faculty requirements and departmental required courses and a minimum overall grade-point-average of 5.0* (for students admitted prior to FW99/00 a minimum grade-point-average of 5.0 over all Science (SC) courses is required).

* *6.0 for the Space and Communication Sciences Stream of Specialized Honours and for Combined Honours with Biology*

All candidates for the BSc degree in all Honours programmes must, through registration in courses at York University or elsewhere deemed creditable towards the BSc degree,

a) satisfy regulations 2, 3, 5 and 6;

b) present a total of at least 120 passed credits of which a minimum of 90 must be earned in Science courses, a minimum of 30 must be earned in one major Science subject area (Specialized Honours programmes only), or a minimum of 18 in each of two major Science subject areas (Combined Honours programmes only), a minimum of 42 must be earned in courses at the 3000 or higher level;

c) satisfy the Faculty academic standards for Honours BSc programmes - see "Academic Standards for Honours BSc Programmes" in section III of the University Undergraduate Calendar;

d) satisfy the programme of study requirements specified below for the declared Honours programme:

Computer Science B.Sc. Requirements

Note: The following degree requirements do not apply to students in the Space and Communication Sciences Stream of Specialized Honours Computer Science; for requirements of that stream, see the "Space and Communication Sciences" Supplemental Calendar and section V of the York University Undergraduate Calendar.

i) All degree candidates must complete the programme core:
   - SC/COSC1020 3.0; SC/COSC1030 3.0;
   - SC/COSC2001 3.0; SC/COSC2011 3.0; SC/COSC2021 3.0.

ii) All degree candidates must comply with general regulation 6 (see above) by completing the following (in addition to SC/COSC1020 3.0 and SC/COSC1030 3.0 from the programme core):
   - SC/MATH1090 3.0; SC/MATH1300 3.0; SC/MATH1310 3.0;
   - 6 credits from SC/BIOL1010 6.0, SC/CHEM1000 6.0, SC/EATS1010 6.0, SC/PHYS1410 6.0 or SC/PHYS1010 6.0; (Note: In this context, SC/COSC1020 3.0 and SC/COSC1030 3.0 satisfy the other half of the 1000-level Science requirement for courses with laboratories.)
   - at least 3 additional credits from SC/BIOL1010 6.0, SC/CHEM1000 6.0, SC/EATS1010 6.0, SC/EATS1010 3.0, SC/EATS1011 3.0, SC/MATH1025 3.0, SC/PHYS1070 3.0, SC/PHYS1410 6.0 or SC/PHYS1010 6.0, SC/BC1800 3.0;
   - 12 general education credits (see "General Education Requirements" in Science section IV).

iii) All degree candidates, in accordance with their declared programmes, must comply with General Regulation 7 or 8 (above) and, in so doing, must also satisfy the course, credit and standing requirements specified below.

iv) All degree candidates must satisfy a breadth requirement in Computer Science by completing 3 credits at the 3000 level from Group A courses in each of four areas:

- theory (second digit of course number is 1),
- hardware (second digit is 2),
- software (second digit is 3) and
- knowledge based (second digit is 4).

Group A courses have odd course numbers; Group B courses have even course numbers.

v) All Honours degree candidates must complete at least 30 credits which are neither computer science nor mathematics.

Note: See the general prerequisites for 2000-, 3000- and 4000-level Computer Science courses (under "Courses of Instruction") for information about cumulative grade-point-average requirements in completed Computer Science courses.

Ordinary Programme

- SC/MATH 2320 3.0
- at least 18 credits from Computer Science courses at the 3000-level satisfying the departmental breadth requirement, for an overall total of at least 33 credits from Computer Science courses (including those within the programme core);
- additional elective credits as required for an overall total of at least 90 credits, including at least 66 credits from Science courses.

Specialized Honours

- SC/MATH2090 3.0; SC/MATH2221 3.0; SC/MATH2320 3.0;
- SC/COSC3101 3.0; SC/COSC4101 3.0 or SC/COSC4111 3.0;
- at least 18 additional credits in Computer Science courses at the 3000 level satisfying the departmental breadth requirement;
- at least 9 additional credits in Computer Science courses at the 4000 level;

- at least 6 additional credits in Computer Science courses at the 3000 or 4000 level, for an overall total of at least 54 credits in Computer Science courses (including those within the programme core);
- additional elective credits as required from an overall total of at least 120 credits, including at least 90 credits from Science courses, at least 42 credits at the 3000 or higher level, and at least 30 credits which are neither computer science nor mathematics.

The Department of Computer Science also offers a Specialized Honours degree stream in Space and Communication Sciences whose degree requirements are specified in a separate entry in section V.

Honours Double Major

- SC/MATH2090 3.0; SC/MATH1025 3.0 or SC/MATH2221 3.0 or SC/MATH2320 3.0;
- at least 15 credits in Computer Science courses at the 3000 level satisfying the departmental breadth requirement;
- at least 12 credits in Computer Science courses at the 4000 level, for an overall total of at least 42 credits in Computer Science courses (including those within the programme core);
- additional credits (including those required for the second major) as required for an overall total of at least 120 credits, including at least 90 credits from Science courses, at least 42 credits at the 3000 or higher level, and at least 30 credits which are neither computer science nor mathematics.

Residence Requirement

In order to qualify for a York University BSc. degree in any Ordinary or Honours programme, a student must have successfully completed a minimum of 30 credits, approved by the major department(s) at York University.

Computer Science Requirements

Breadth requirement

We have partitioned our courses into four areas. Students must take at least one 3000-level course identified as a **Group A** course from each area. **Group A** courses have **odd course numbers**.

4000-level courses are also partitioned into the same four areas but they are not further partitioned into groups A and B. Thus, whether the course number is even or odd has no significance for 4000-level courses.

The four areas are as follows:

Theory – Course numbers COSC31xx 3.0, COSC41xx 3.0; topics: algorithms, data structures and complexity, automata and computability, program verification, scientific and numerical computing.

Hardware – Course numbers COSC32xx 3.0, COSC42xx 3.0; topics: digital logic, architecture and data communication networks.

Software – Course numbers COSC33xx 3.0, COSC43xx 3.0; topics: programming languages, software systems and operating systems.

Knowledge-Based Computing – Course numbers COSC34xx 3.0, COSC44xx 3.0; topics: artificial intelligence, expert systems, logic programming, databases, simulation, machine learning, robotics and computer vision.

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 -- restricted to SCS stream students
- 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 programme 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 Programme Checklists.

The indication of first year, second year, etc., indicates the year of study for normal progress.

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 programme, or elective requirements

2000-level – second year

- COSC2001 3.0, COSC2011 3.0, COSC2021 3.0
- MATH2090 3.0
- MATH2221 3.0 **and** MATH2320 3.0 - Specialized Honours programme; MATH2221 3.0 **or** MATH2320 3.0 all other programmes
- 12 to 15 additional credits satisfying general education, faculty, second major programme, or elective requirements

3000-level – third year

- 12 COSC credits at the 3000-level satisfying the breadth requirement including COSC3101 3.0 for the Specialized Honours programme (except SCS)
- 9 additional COSC credits for the Specialized Honours Programme; or 3 additional COSC credits at the 3000-level for Combined Honours, Honours Major and Honours Minor programmes

- 9 to 15 credits additional credits satisfying general education, faculty, second major programme, or elective requirements

4000-level – fourth year, honours programmes only

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

Prerequisites for Computer Science Courses[1]

It is absolutely essential that students fulfill the 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
2000-Level		

General Prerequisites:

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

[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.*

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	Fundamentals of Unix, C and C++	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
- completed MATH1300 3.0 and MATH1310 3.0
- completed one of MATH2090 3.0, MATH2221 3.0 or MATH2320 3.0
- have a cumulative GPA of 4.5 or better over all completed Computer Science courses.

		Prerequisites
Theory Courses - Group A		
COSC3101 3.0	Design and Analysis of Algorithms including students may enrol 3.0 MATH2320 3.0)	General prerequisites MATH2320 3.0 (SCS without MATH2320 or concurrently with
COSC3111 3.0	Intro. to Program Verification including MATH2090 3.0	General prerequisites
COSC3121 3.0	Intro. to Numerical Computations I including MATH2221 3.0	General prerequisites
Theory Courses - Group B		
COSC3122 3.0	Intro. to Numerical Computations II MATH2270 3.0	COSC3121 3.0;

Hardware Courses - Group A

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

Hardware Courses - Group B

COSC3212 3.0	Computer Networks	COSC3211 3.0

Software Courses - Group A

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
COSC3321 3.0	Operating System Fundamentals	General prerequisites including COSC2021 3.0; COSC2031 3.0
COSC3331 3.0	Object-Oriented Programming and Design	General prerequisites

Knowledge-Based Computing - Group A

COSC3401 3.0	Intro. to Symbolic Computation	General prerequisites including MATH2090 3.0

Knowledge-Based Computing - Group B

COSC3402 3.0	Intro. to Concepts of Artificial Intell.	COSC3401 3.0; MATH2320 3.0
COSC3408 3.0	Simulation of Discrete Systems	MATH2560 3.0

COSC3418 3.0	Simulation of Continuous Systems	MATH2560 3.0
COSC3421 3.0	Introduction to Database Systems	ITEC/COSC2011 3.0
COSC3461 3.0	Human-Computer Interaction	ITEC/COSC2011 3.0

Other Courses:

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

4000-Level

General Prerequisites:

- completed COSC2001 3.0; COSC2011 3.0; COSC2021 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

Specific Prerequisites

Theory Courses

COSC4101 3.0	Advanced Data Structures	COSC3101 3.0; MATH2320 3.0
COSC4111 3.0	Automata and Computability	COSC3101 3.0; MATH2320 3.0

Hardware Courses

COSC4201 3.0	Computer Architecture	COSC3201 3.0; COSC3321 3.0
COSC4211 3.0	Performance Eval. of Computer Systems	COSC3211 3.0; COSC3408 3.0
COSC4242 3.0	Signals and Systems	COSC3121 3.0 (MATH3241 3.0)

Software Courses

COSC4301 3.0	Programming Language Design	COSC3301 3.0
COSC4302 3.0	Language Processors	COSC3301 3.0; COSC2031 3.0
COSC4311 3.0	System Development	COSC3311 3.0; COSC3111 3.0 or COSC3321 3.0
COSC4321 3.0	Operating System Design	COSC3321 3.0
COSC4331 3.0	Computer Graphics	MATH2221 3.0
COSC4341 3.0	Interactive System Design	COSC3461 3.0
COSC4351 3.0	Real-Time Systems Theory	COSC3111 3.0 or COSC3311 3.0 or COSC3321 3.0
COSC4352 3.0	Real-Time Systems Practice	COSC3301 3.0 or COSC3311 3.0 or COSC3321 3.0
COSC4361 3.0	Human-Computer Communication	COSC3461 3.0

Knowledge-Based Computing

COSC4401 3.0	Artificial Intelligence	COSC3402 3.0
COSC4402 3.0	Logic Programming	COSC3401 3.0; COSC3101 3.0 or COSC3111 3.0
COSC4411 3.0	Database Management Systems	COSC3412 3.0
COSC4421 3.0	Introduction to Robotics	MATH2221 3.0
COSC4422 3.0	Computer Vision	COSC3121 3.0 (MATH3241 3.0)

Other Courses:

COSC4001 6.0	Space 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

Checklist[1]

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	9
	MATH2320 3.0			3
3000-level	One Group A course (odd numbered) from each area			
	Theory COSC31_____ 3 0	Software	COSC33____ 3.0	6
	Hardware COSC32_____ 3.0	Knowledge	COSC34____ 3.0	6
	Two more courses	COSC3_____ 3.0	COSC3_____ 3.0	6

Faculty Requirements

General Education Courses		_____	_____	12
One of	BIOL1010 6.0	CHEM100 6.0	EATS1010 6.0	
	PHYS1010 6.0	PHYS1410 6.0		6
At least 3 additional credits from the following:				
	BIOL1010 6.0	CHEM1000 6.0	EATS1010 6.0	
	EATS1010 3.0	EATS1011 3.0	BC1800 3.0	
	MATH1025 3.0	PHYS1010 6.0	PHYS1070 3.0	
	PHYS1410 6.0			3
12 more SC credits	_____	_____		6
	_____	_____		6

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

12 more credits	_____	_____	6
	_____	_____	6
		Total credits	90

BSc Specialized Honours Degree

Checklist[1]

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	9
	MATH2090 3.0	MATH2221 3.0	MATH2320 3.0	9
3000-level	One Group A course (odd numbered) from each area			
	Theory COSC3101 3.0	Software COSC33____ 3.0		6
	Hardware COSC32_____ 3.0	Knowledge COSC34____ 3.0		6
	Three more courses:			
	COSC3_____ 3.0	COSC3_____ 3.0	COSC3_____ 3.0	9
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 Courses	_____	_____	12
One of	BIOL1010 6.0, CHEM1000 6.0, EATS1010 6.0, PHYS1010 6.0, PHYS1410 6.0		6
At least 3 additional credits from the following:			
	BIOL1010 6.0	CHEM1000 6.0	EATS1010 60
	EATS1010 3.0	EATS1011 3.0	BC1800 3.0

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

MATH1025 3.0	PHYS1010 6.0	PHYS1070 3.0	PHYS1410 6.0	3
9 more SC credits[2]	_____	_____	_____	9
18 more credits	_____	_____	_____	9
	_____	_____	_____	9
			Total credits	120

[2] *At least three credits in total from the categories "9 more SC credits" and "18 more credits" must be at the 3000- or 4000-level (for an overall total of 42 credits at 3000- or 4000-level). If MATH1025 3.0 is taken, then 12 credits are required in this category, as MATH1025 3.0 is considered equivalent to MATH2221 3.0 for Computer Science requirements and is counted twice in the checklist.*

BSc Honours Double Major Degree
(formerly Combined Honours before 99/00)

Checklist[1]

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	9
	MATH2090 3.0	MATH2221 3.0 **or** MATH2320 3.0		6
3000-level	One Group A course (odd numbered) from each area			
	Theory COSC31_____ 3.0	Software COSC33____ 3.0		6
	Hardware COSC32_____ 3.0	Knowledge COSC34____ 3.0		6
	One more course	COSC3_____ 3 0		3
4000-level	Four courses	COSC4_____ 3.0	COSC4_____ 3.0	6
		COSC4_____ 3.0	COSC4_____ 3.0	6

Faculty Requirements

General Education courses:		_____	_____	12
One of	BIOL1010 6.0	CHEM1000 6.0	EATS1010 6.0	
	PHYS1010 6.0	PHYS1410 6.0		6
At least 3 additional credits from the following				
	BIOL1010 6.0	CHEM1000 6.0	EATS1010 6.0	
	EATS1010 3.0	EATS1011 3.0	BC1800 3.0	
	MATH1025 3.0	PHYS1010 6.0	PHYS1070 3.0	
	PHYS1410 6.0			3

Other Honours Subject and Other Courses

---

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

42 more credits[2]

_____	_____	_____	_____	_____	
_____	_____	_____	_____	_____	
_____	_____	_____	_____	_____	42
				Total credits	120

[2] *If MATH1025 3.0 is taken, then 45 credits are required in this category, as MATH1025 3.0 is considered equivalent to MATH2221 3.0 for Computer Science requirements and is counted twice in the checklist.*

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				6
		CHEM1000 6.0 **or** EATS1010 6.0				6
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	3
		EATS2030 3.0 **or**		PHYS1070 3.0 **or**	PHYS2060 3.0	
3000-level		COSC3121 3.0		COSC3211 3.0	COSC3321 3.0	9
		COSC/EATS/PHYS/3001 1.0				1
		EATS/PHYS3280 3.0				3
		PHYS3050 3.0		PHYS3250 3.0		6
	One of	COSC3311 3.0 **or**		COSC3331 3.0 **or**	COSC3401 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	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 3.0	6
	or PHYS4450 3.0					

Faculty Requirements

General Education Courses			12
	Total credits		122 or 123

BA Ordinary Degree

Checklist[1]

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	9
	MATH2320 3.0			3
3000-level	One Group A half course (odd numbered) from each area			
	Theory COSC31_____ 3.0	Software	COSC33____ 3.0	6
	Hardware COSC32_____ 3.0	Knowledge	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 (Extra course required for breadth if not satisfied.)				
	One of HUMA_____ 9.0	**or**	SOSC_____ 9.0	9
Electives	18 credits outside COSC requirements			
	_____	_____	_____	18
Additional course	_____			3

[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 programme.*

Total Credits	90

BA Honours Major Degree

Checklist[1]

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	9
	MATH2090 3.0	MATH2221 3.0 **or**	MATH2320 3.0	6
3000-level	One Group A half course (odd numbered) from each area			
	Theory COSC31_____ 3.0		Software COSC33_____ 3.0	6
	Hardware COSC32_____ 3.0		Knowledge COSC34_____ 3.0	6
	One more half course: COSC3_____ 3.0			3
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 (Extra course required for breadth if not satisfied.)				
	One of HUMA_____ 9.0	**or**	SOSC_____9.0	9
Electives	18 credits outside COSC requirements			

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

	_____	_____	_____	18
Upper level	3000-level		_____	3
	4000-level full		_____	6
Additional courses	_____	_____	_____	
	_____			12
		Total Credits		120

BA Honours Minor Degree

Checklist[1]

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	9
	MATH2090 3.0	MATH2221 3.0 **or**	MATH2320 3.0	6
3000-level	One Group A half course (odd numbered) from each area			
	Theory COSC31_____ 3.0		Software COSC33_____ 3.0	6
	Hardware COSC32_____ 3.0		Knowledge COSC34_____ 3.0	6
	One more half course: COSC3_____ 3.0			3
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 (Extra course required for breadth if not satisfied.)				
	One of HUMA_____ 9.0	**or**	SOSC_____9.0	9

Honours Major subject

---

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

3000-level	(to satisfy upper level requirement)	_____		3
4000-level	(to satisfy upper level requirement)			
		_____	_____	12
Additional courses	_____	_____	_____	
	_____	_____	_____	
	_____	_____	_____	30
			Total Credits	120

BA Specialized Honours Degree

Checklist[1]

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	9
	MATH2090 3.0	MATH2221 3.0	MATH2320 3.0	9
3000-level	One Group A half course (odd numbered) from each area			
	Theory COSC31_____ 3.0		Software COSC33_____ 3.0	6
	Hardware COSC32_____ 3.0		Knowledge COSC34_____ 3.0	6
	Three more courses:			
	COSC3_____ 3.0	COSC3_____ 3.0	COSC3_____ 3.0	9
4000-level	Four courses	COSC4101 3.0 **or** COSC4111 3.0		3
		COSC4_____ 3.0		3
		COSC4_____ 3.0	COSC4_____ 3.0	6
	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

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

	2000-level (Extra course required for breadth if not satisfied)	
	One of HUMA_____ 9.0 **or** SOSC_____ 9.0	9
Electives	3 courses outside COSC requirements	
	_____ _____ _____	18
Upper level	4000-level[2] _____	6
	Total Credits	120

[2] *Must be 4000-level only if you have taken less than 18 Computer Science credits at the 4000-level. Otherwise the course can be at any level.*

BA Honours Double Major Degree

Checklist[1]

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	9
	MATH2090 3.0	MATH2221 3.0 **or**	MATH2320 3.0	6
3000-level	One Group A half course (odd numbered) from each area			
	Theory COSC31_____ 3.0		Software COSC33_____ 3.0	6
	Hardware COSC32_____ 3.0		Knowledge COSC34_____ 3.0	6
	One more half course	COSC3_____ 3.0		3
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 (Extra course required for breadth if not satisfied above)				
	One of HUMA_____ 9.0	**or**	SOSC_____ 9.0	9

Other Honours Major Subject and Other Courses

---

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

3000-level	(to satisfy upper level requirement)	_____		3
4000-level	(to satisfy upper level requirement)			
		_____	_____	12
Additional courses	_____	_____	_____	
	_____	_____	_____	
_____	_____	_____	_____	30

Total Credits 120