Instructor Information and Office hours

Dr. Rami Sabouni; Office: MC 4032; Email: rami.sabouni@carleton.ca. Office hours are posted on cuLearn.

TA Information and Office hours

TA information will be posted on cuLearn.

Course Number and Calendar Description

SYSC 2006 [0.5 credit]
Foundations of Imperative Programming
Modular programming with a procedural language. Compilation and linking, libraries. Memory management and object lifetimes: static allocation, automatic allocation in stack frames, dynamic allocation from the heap. Introduction to data structures: dynamic arrays, linked lists. Collections: lists, stacks, queues. Introduction to recursion. Precludes additional credit for SYSC 1102, SYSC 2002 and COMP 2401. Lectures six hours a week, laboratory four hours a week.

Prerequisites

ECOR 1606 or SYSC 1005 are the prerequisites for SYSC 2006. Prerequisite waivers will not normally be granted. Students who have not received credit for ECOR 1606 or SYSC 1005 must withdraw from SYSC 2006 by the last date for registration in Winter term courses. Students who received DEF in the Fall 2018 offering of SYSC 1005 or ECOR 1606 should speak to the course instructor as soon as possible about their eligibility to remain in SYSC 2006; i.e., before the deferred exam is written.

Students in the B.Sc. Honours in Applied Physics who have completed COMP 1005 or COMP 1405 will be considered to have satisfied the prerequisite for SYSC 2006.

Course Objectives

After completing this course, students should:

- understand the concepts that underlie most imperative programming languages and be able to use this knowledge to help them learn new languages;
- have developed a "mental model" of computation; in other words, learned how to reason about and visualize the execution of program code;
• understand the design and application of two fundamental data structures: the dynamic (resizable) array and the pointer-based singly-linked list;
• be able to construct simple recursive functions;
• be prepared to undertake a course that provides a thorough introduction to object-oriented programming principles.

Learning Outcomes

By the end of this course, a student should:

1. understand several constructs provided by imperative programming languages that determine the flow of control when a program is executed;
2. understand several constructs provided by imperative programming languages for structuring data;
3. be able to trace short programs written in an imperative programming language and
   • explain what happens, step-by-step, as the computer executes each statement;
   • visualize how code execution changes the program's state; in other words, draw diagrams that depict the program's global variables, its activation frames (containing function parameters and local variables) and memory that has been allocated from the heap and is accessed through pointers.
4. be able to design, code, test and visualize the execution of functions that operate on two fundamental data structures: the dynamic (resizable) array and the pointer-based singly-linked list.
5. be able to specify simple recursive algorithms, convert these algorithms into recursive functions, and visualize the execution of recursive code.

Graduate Attributes (GA's)

The Canadian Engineering Accreditation Board requires graduates of engineering programs to possess 12 attributes. Activities related to the learning outcomes listed here are intended to develop students' competence in GA 1 (a knowledge base for engineering: demonstrated competence in specialized engineering knowledge appropriate to the program - programming and algorithms). Data obtained from exam questions related to the learning outcomes will be collected to assess students' progress towards achieving GA 1.

By working on programming exercises ("lab work" and other assigned exercises), students will develop skills related to elements of GA 5 (Use of Engineering Tools) and GA 7 (Communication skills). Students' progress towards achieving GAs 5 and 7 will not be assessed in this course.

In addition, activities related to the learning outcomes are intended to introduce students to techniques for problem analysis and design of complex engineering problems in the context of software development, and to prepare students to undertake learning activities in subsequent courses that develop competence in GA 2 (Problem Analysis) and GA 4 (Design). Students' progress towards achieving GAs 2 and 4 will not be assessed in this course.
Course Web Site

This course uses cuLearn, Carleton's learning management system. To access your courses on cuLearn, go to carleton.ca/culearn.

For help and support, go to carleton.ca/culearnsupport/students. Any unresolved questions can be directed to Computing and Communication Services (CCS) by phone at 613-520-3700 or via email: ccs_service_desk@carleton.ca.

Textbook and References

- **How to Think Like a Computer Scientist - C Version, Version 1.09** (January, 2018) or newer, Allen Downey and Thomas Scheffler.

  This book is available under a license (GNU General Public License, Version 2) that allows readers to freely copy and distribute the text. A PDF copy of the most recent version can be downloaded from Prof. Scheffler's *Programmieren in C* Web page:


  Two versions are available: one written in English, the other in German.

- **A Tour of Go** website, [https://tour.golang.org/list](https://tour.golang.org/list).
- **The Go Programming Language: Documentation** website, [https://golang.org/doc/](https://golang.org/doc/).

  One of the authors (Ritchie) was the original developer of C. The second edition of this book describes the first standard version of C, which is often referred to as ANSI C (C89) or ISO C (C90). The book has not been updated to reflect newer versions of the language, such as C99 or C11 (the current standard for the language); however, it's still widely used and is regarded by many as being an authoritative reference on C.


Links to Software, Libraries and Additional Resources

- **C Tutor** (visit [pythontutor.com](http://pythontutor.com))

  Learning how to trace and explain the execution of short programs is an important learning outcome in this course. C Tutor is a free Web-based tool that helps us visualize what happens as the computer executes each line of a program's source code, step-by-step. We've
used the Python visualization tool hosted at this site for several years in SYSC 1005. Support for visualizing C program execution was added recently, and although it has some "rough edges", we've used it in recent offerings of the course and feedback from students has been positive.

- **Software**

Pelles C, the C programming environment used in this course, is free. Version 8.00 is installed on our lab computers. Version 9.00 was released after our lab computers were set up for the current academic year. You can use either version.

Pelles C 9.00 can be downloaded from this website: [smorgasbordet.com/pellesc/](http://smorgasbordet.com/pellesc/). Pelles C 8.00 and 9.00 can be downloaded from this website: [www.pellesc.de/](http://www.pellesc.de/).

We are unable to provide support for students who prefer to use Mac OS X or Linux. During the labs, only Pelles C projects will be graded. If you decide to develop C code using another Windows IDE or using OS X or Linux tools, it is your responsibility to transfer your code to a Pelles C project and verify that it executes correctly before you demonstrate it to a TA.

**Evaluation and Grading Scheme**

Students will be evaluated primarily by means of a midterm test and a final exam. In addition, the marks assigned for lab work contribute towards the final grade.

To pass the course, students must pass the final examination. For students who pass the final exam, a numeric mark out of 100 will be calculated by weighting the course components according to Scheme 1:

<table>
<thead>
<tr>
<th>Component</th>
<th>Scheme 1</th>
<th>Scheme 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lab work</td>
<td>0%</td>
<td>15%</td>
</tr>
<tr>
<td>Midterm test</td>
<td>30%</td>
<td>25%</td>
</tr>
<tr>
<td>Final exam</td>
<td>70%</td>
<td>60%</td>
</tr>
</tbody>
</table>

This mark will be converted to a letter grade, using the table of percentage equivalents shown in *Section 2.3 of the Academic Regulations of the University*.

If the grade under Scheme 1 is D+ or lower, this will be your final grade. If the letter grade under Scheme 1 is C- or higher, a second numeric mark will be calculated using Scheme 2's component weighting, and this mark will be converted to a second letter grade. Your final grade will be the higher of the two letter grades.

**Early Feedback**

*See Section 2.2.1 of the Academic Regulations of the University.*
The weekly lab exercises will normally be graded during the lab period. Outside of the scheduled labs, you can obtain feedback during office hours or by making an appointment to see your instructor.

**Lab Periods**

Attendance at the scheduled laboratory periods is mandatory, and attendance will be taken. During the labs you will work on short programming exercises that are intended to help you understand concepts that have been introduced in the lectures. You will normally be required to demonstrate your lab work by the end of the lab period, as indicated in that week’s lab "handout”.

When you demonstrate your lab work, you may be asked by a TA to provide a detailed explanation of your solution to one of the exercises (e.g., discuss your design decisions, explain how you would modify your code to reflect different requirements, etc.) Your explanation will contribute to your grade for that week's lab.

During the lab period, your work will be graded satisfactory, marginal, or unsatisfactory.

- **Satisfactory** means that you were present at the lab and made reasonable progress towards completing the exercises. Note that you do not have to finish all the exercises to receive a satisfactory grade.
- **Marginal** means that you made some progress towards completing the exercises, but your solutions were not sufficiently complete to warrant a satisfactory grade. This grade indicates that you may be falling behind and should take steps to remedy this situation.
- **Unsatisfactory** means that you made little or no progress towards completing the lab exercises. This indicates that you are likely having difficulty understanding important concepts and should seek help from your instructor as soon as possible. You will also receive unsatisfactory if it is apparent to the TA that you did not do enough of the lab work on your own; that is, you relied on your colleagues to explain the exercises and provide solutions (approach, algorithms or code).

For each satisfactory or marginal grade, you will receive 1/1 towards the lab component of the course. Each unsatisfactory grade will receive 0/1.

You must attend the lab section in which are enrolled. If you demonstrate your work in a different lab section, you will receive 0 for that lab.

Students cannot "make up" missed labs by attending an alternate lab section. Here's what you should do if you miss a lab because of illness, a medical appointment, job interview, etc.

- **If you have documentation that confirms the reason for your absence (e.g., a medical certificate, appointment card or email from a doctor's office, or other appropriate supporting documentation):** Do the missed lab work on your own time. No later than three working days after your lab, upload your lab work to cuLearn and send your instructor an email that clearly indicates your lab section and which lab you missed. Attach a scanned
copy of the supporting documentation or make arrangements show the document to your instructor in person. A TA will be assigned to review your lab work and assign a grade.

- **If you don't have supporting documentation:** Do the missed lab work on your own time. No later than three working days after your lab, upload your lab work to cuLearn and send your instructor an email that clearly indicates your lab section and which lab you missed. A TA will be assigned to review your lab work and assign a grade. **Students can receive credit for at most one missed lab without providing supporting documentation. For all subsequent missed labs, supporting documentation must be provided; otherwise, you will receive 0 for those labs.** We strongly recommend that you not use this accommodation to "skip" a lab so that you can study for a test. You should instead complete the exercises ahead of time and demonstrate your work at the start of the lab; you can then use the remainder of the lab session to study.

If you miss a lab for reasons related to disabilities, pregnancy or religious obligations, please contact your instructor immediately to arrange appropriate accommodations. (See Academic Accommodations, towards the end of this document). For example, a student who is registered with the Paul Menton Centre and who misses multiple labs for reasons related to the disability, will not be required to provide supporting documentation over and above the documentation that was originally supplied to the PMC. Arrangements will be made so that it is possible for the student to receive credit for all labs absences that are related to the disability.

Serious long-term illness will be dealt with on an individual basis; in these circumstances, please contact your instructor to discuss appropriate arrangements.

Portions of the designs and code from any lab may be reused and refined in subsequent labs, and doing the labs is the best way to learn the course material and prepare for the exams, so students are encouraged not to "write off" any particular lab just because of its relatively low weight in the overall grading scheme.

Students are responsible for backing up their lab work before they leave the lab; for example, we recommend that you copy your files to a USB flash drive or to a cloud-based file hosting service; e.g., Google Drive, Dropbox, OneDrive, etc. **Requests to attend an alternate lab section because you don't have the files from previous labs that are required for the current lab, will not be approved.**

Students can use the Systems and Computer Engineering undergraduate computer labs and the computer labs operated by the Office of the Dean of Engineering whenever the Mackenzie Building, Minto CASE and the Canal Building are open, except for those times when labs are reserved for specific courses.

**Midterm Test**

There will be one closed-book midterm test, which will be held in class on May 23th, 2019. Computers will not be used during the midterm test.
Students who are unable to write the midterm test because of illness or other circumstances beyond their control (e.g., family or religious obligations) should contact their instructor to request accommodation for the missed test. These requests must be made no later than 3 working days after the test date, and must fully supported by appropriate documentation (in cases of illness, a medical certificate is required). For more information, see the Academic Regulations of the University, Section 2.6, Deferred Term Work.

Requests for accommodation because of poor performance on the midterm test will not be considered. So, if you are ill on the day of the midterm test, don't write the test and later claim that your performance was impaired because you were unwell. You are better off to miss the test obtain a medical certificate and request accommodation.

Final Exam

A closed-book final exam will be held during the University's June examination period. Computers will not be used during the final exam. All students are eligible to write the final examination, regardless of the marks they received during the term.

Students who miss the final exam because of illness or other circumstances beyond their control may apply to write a deferred examination. For more information, see the Academic Regulations of the University, Sections 2.5, Deferred Final Examinations, 2.5.1, Missed Deferred Examinations, and 2.5.2, Early Departure from Final Examinations.

The final examination is for evaluation purposes only and will not be returned to students. You will be able to make arrangements with the instructor or with the department office to see your marked final examination after the final grades have been made available.

General Regulations

Copyright on Course Materials:

The materials created for this course (including the course outline and any slides, posted notes, labs, project, assignments, quizzes, exams and solutions) are intended for personal use and may not be reproduced or redistributed or posted on any web site without prior written permission from the author(s).

Attendance:

Students are expected to attend all lectures and lab periods. The University requires students to have a conflict-free timetable. For more information, see the current Undergraduate Calendar, Academic Regulations of the University, Section 1.2, Course Selection and Registration and Section 1.5, Deregistration.

Health and Safety:

Every student should have a copy of our Health and Safety Manual. A PDF copy of this manual is available online: http://sce.carleton.ca/courses/health-and-safety.pdf

Deferred Term Work:
Students who claim illness, injury or other extraordinary circumstances beyond their control as a reason for missed term work are held responsible for immediately informing the instructor concerned and for making alternate arrangements with the instructor and in all cases this must occur no later than three (3.0) working days after the term work was due. The alternate arrangement must be made before the last day of classes in the term as published in the academic schedule. For more information, see the current Undergraduate Calendar, Academic Regulations of the University, Section 2.6, Deferred Term Work.

Appeal of Grades:
The processes for dealing with questions or concerns regarding grades assigned during the term and final grades is described in the Undergraduate Calendar, Academic Regulations of the University, Section 2.7, Informal Appeal of Grade and Section 2.8, Formal Appeal of Grade.

Academic Integrity:
Students should be aware of their obligations with regards to academic integrity. Please review the information about academic integrity at: https://carleton.ca/registrar/academic-integrity/. This site also contains a link to the complete Academic Integrity Policy that was approved by the University's Senate.

Plagiarism:
Plagiarism (copying and handing in for credit someone else's work) is a serious instructional offense that will not be tolerated.

Academic Accommodation:
You may need special arrangements to meet your academic obligations during the term. You can visit the Equity Services website to view the policies and to obtain more detailed information on academic accommodation at http://www.carleton.ca/equity/ For an accommodation request, the processes are as follows:

- **Pregnancy obligation:**
  write to me with any requests for academic accommodation during the first two weeks of class, or as soon as possible after the need for accommodation is known to exist. For more details see https://carleton.ca/equity/wp-content/uploads/Student-Guide-to-Academic-Accommodation.pdf

- **Religious obligation:**
  write to me with any requests for academic accommodation during the first two weeks of class, or as soon as possible after the need for accommodation is known to exist. For more details see https://carleton.ca/equity/wp-content/uploads/Student-Guide-to-Academic-Accommodation.pdf

- **Academic Accommodations for Students with Disabilities:**
The Paul Menton Centre for Students with Disabilities (PMC) provides services to students with Learning Disabilities (LD), psychiatric/mental health disabilities, Attention Deficit Hyperactivity Disorder (ADHD), Autism Spectrum Disorders (ASD), chronic medical conditions, and impairments in mobility, hearing, and vision. If you have a disability requiring academic accommodations in this course, please contact PMC at 613-520-6608 or pmc@carleton.ca for a formal evaluation. If you are already registered with
the PMC, contact your PMC coordinator to send me your Letter of Accommodation at the beginning of the term, and no later than two weeks before the first in-class scheduled test or exam requiring accommodation (if applicable). After requesting accommodation from PMC, meet with me to ensure accommodation arrangements are made. Please consult https://carleton.ca PMC/students/dates-and-deadlines/ for the deadline to request accommodations for the formally-scheduled exam (if applicable).

- **Survivors of Sexual Violence:**
  As a community, Carleton University is committed to maintaining a positive learning, working and living environment where sexual violence will not be tolerated, and where survivors are supported through academic accommodations as per Carleton’s Sexual Violence Policy. For more information about the services available at the university and to obtain information about sexual violence and/or support, visit: https://carleton.ca/sexual-violence-support/.

- **Accommodation for Student Activities:**
  Carleton University recognizes the substantial benefits, both to the individual student and for the university, that result from a student participating in activities beyond the classroom experience. Reasonable accommodation must be provided to students who compete or perform at the national or international level. Please contact your instructor with any requests for academic accommodation during the first two weeks of class, or as soon as possible after the need for accommodation is known to exist. For more details, see https://carleton.ca/senate/wp-content/uploads/Accommodation-for-Student-Activities-1.pdf
Topics

Most of the code examples will be written in C, but for comparison purposes code written in other languages (e.g., Go and Python) will be presented. C will be the primary language used in the labs, but some lab exercises or post-lab exercises (to be completed on your own time) may use Go.

- Fundamental elements of imperative programming languages: types, variables, expressions, control flow: conditional statements, iteration (loops), functions (subroutines/procedures).
- Function calls and parameter-passing mechanisms. Visualizing program state by drawing memory diagrams containing activation frames (activation records) that depict function parameters and local variables.
- Structuring data: arrays.
- Motivation for modular programming. Modules: interface vs. implementation.
- Pointers. Depicting pointers in memory diagrams.
- Structuring data: structures.
- Introduction to dynamically-allocated memory and the heap. Drawing memory diagrams that illustrate how parameters and local variables in activation frames can point to memory blocks allocated on the heap. Memory leaks. Dynamically allocated structures.
- Dynamically-allocated arrays, dynamic arrays.
  - Case study: C implementation of a list collection using a dynamic array.
  - Case study: Go slices.
- Structuring data: linked lists. Drawing memory diagrams to understand the fundamental operations on singly-linked lists. Implementing linked lists. Applications of linked lists.
- Introduction to recursion.

Edited: December 24, 2018