Carleton University
Department of Systems and Computer Engineering
SYSC 2006 A - Foundations of Imperative Programming – Early Summer 2018
Course Outline

Instructor

Dr. Lynn Marshall; Office: ME 4230; Email: lynnmar@sce.carleton.ca. Office hours are posted on cuLearn.

Undergraduate Calendar Course 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.
Prerequisite(s): ECOR 1606 or SYSC 1005.
Lectures three hours a week, laboratory two hours a week.

Course Aims

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;
- understand how memory is managed by an executing program, and demonstrate this knowledge pictorially;
- understand different designs for simple abstract linear collections such as lists (vectors), queues and stacks;
- 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 be able to:

1. trace short C programs 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 contents of the program's global variables, its activation frames (containing function arguments and local variables) and memory that has been allocated from the heap and is accessed through pointers.

2. design, code and test functions that operate on two fundamental data structures: the dynamic (resizable) array and the pointer-based singly-linked list.

3. describe, from a client-side perspective, the operations provided by some linear abstract collections; e.g., lists (vectors), queues and stacks.

4. implement these collections; i.e., given the specification of a collection's operations and a description of its underlying data structure, implement the data structure and the algorithms that provide the required operations.

5. specify simple recursive algorithms, convert these algorithms into recursive functions, and draw memory diagrams to explain their execution.

Graduate Attributes (GAs)

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 - specialized engineering knowledge appropriate to the program). Data obtained from exam questions related to learning outcomes 1 and 3 will be collected to assess students' progress towards achieving GA 1.

In addition, activities related to learning outcomes 2, 4 and 5 are intended to prepare students to undertake learning activities that develop competence in GA 4 (design solutions for complex, open-ended engineering problems) in subsequent courses.

Prerequisite

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 Early Summer term courses. Students who received DEF in the Winter 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.

cuLearn

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.

Required Textbooks

- *Programming in C*, zyBooks (on-line textbook). Details on subscribing on cuLearn.

Here is how your zyBooks mark will be calculated. Your percentage complete of each chapter (1 to 14) is calculated for the non-optional sections by taking the average of your percentage complete of the participation activities (orange) and the challenge activities (blue) as of the end of term, i.e. 11:55pm June 19th.

Each of chapters 1 to 8 is worth 0.5 marks multiplied by your percentage complete, and each of chapters 9 to 14 is worth 1.0 mark multiplied by your percentage complete.

In addition, if all of chapters 1 through 8 are at least 90% complete by 11:55pm May 29th, you receive 0.5 bonus marks. Finally, if all 12 chapters are at least 95% complete by 11:55pm June 19th, you receive 2 bonus marks.

Thus, you can earn a maximum of 12.5 marks (out of 10). Your zyBooks mark counts as either 5 or 10% of your grade (see grading scheme, below).

Optional Textbooks

- *How to Think Like a Computer Scientist - C Version*, Allen Downey and Thomas Scheffler, 2012.

  This book is available under a license 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:

  http://prof.beuth-hochschule.de/scheffler/lehre/programmieren-in-c/

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

  This book provides a less comprehensive introduction to programming and C than *Programming in C*; specifically, it doesn't cover the material taught in the last half of SYSC 2006.

- Optional Reference: *The C Programming Language*, *Second Edition*, Brian W. Kernighan and Dennis Ritchie, Prentice Hall, 1988, ISBN-10: 0131103628, ISBN-13: 9780131103627.
  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.

Web-based Resources

C Tutor (visit 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's still classified as experimental, we've used it in recent offerings of the course and feedback from students has been positive.

Software

Pelles C Version 8.00, the C programming environment used in this course, is free. If you want to install this software on your own computer, you can download it from these Web sites:

smorgasbordet.com/pellesc/

www.christian-heffner.de/

Only the Pelles C Setup program needs to be downloaded and executed. You don't need to download the Add-In SDK Setup program.

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.

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: sce.carleton.ca/courses/health-and-safety.pdf.

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 completing the zyBooks and lab work will 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:

Component	**Scheme 1**	**Scheme 2**
zyBooks	5%	10%
Lab work	0%	10%
Midterm test	30%	25%
Final exam	65%	55%

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.

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 particular 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 were absent from the lab period, or you attended but 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 you do not demonstrate your work **or 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 s*atisfactory* or *marginal* grade, you will receive 1/1 towards the lab component of the course. Each *unsatisfactory* grade will receive 0/1.

You must normally attend the lab section in which are enrolled. **If you demonstrate your work in a different lab section without obtaining prior permission from the course instructor to do so, you will receive 0 for that lab.**

If you know in advance that you are unable to attend a lab; for example, because of a job interview or a medical appointment, please email your instructor before the lab, if possible. If you are absent from a lab because of illness or other unexpected reason, you must email your instructor before the start of your next lab period. If possible, we will try to arrange for you to "make up" the missed lab by attending another lab section; otherwise, it will be up to you to do the missed lab work on your own time, and you must make arrangements with your instructor to demonstrate your completed work in a timely manner in order to receive credit for the lab. Students may be asked to provide appropriate documentation to confirm the reason for your absence before requests for accommodation are considered. Under no circumstances will students be allowed to "skip" a scheduled lab so that they 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.

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 normally 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.

Exams

There will be one closed-book midterm test, which will be held during a lecture. 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 make arrangements to write a deferred 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 and request permission to write the deferred test, by following the procedure outlined earlier.

A closed-book final exam will be held during the University's June examination period. Computers will **not** be used during the final exam.

The *Academic Regulations of the University* permit instructors to specify requirements that must be satisfied for students to be eligible to write the final examination or, where circumstances warrant, the deferred final examination.

- 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. The granting of a deferred exam requires that the student has performed satisfactorily in the course, excluding the final exam. Students who attended at least 60% of the labs and wrote the midterm test will be deemed to have performed satisfactorily in the course (regardless of their marks in the lab and midterm components) when their applications for a deferral of the final examination are considered. For more information, see the *Academic Regulations of the University*, Section 2.2, *The Course Outline*; Section 2.3, *Standing in Courses/Grading System*; and Section 2.5, *Deferred 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 your instructor to see your marked final examination after the final grades have been made available. Your exam will not be remarked during this meeting and solutions to the exam questions will not be provided.

Appeal of Grade

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

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.

Intellectual Property

Classroom teaching and learning activities, including lectures, labs, discussions, presentations, etc., by both instructors and students, are copy protected and remain the intellectual property of their respective author(s). All course materials, including course outlines, lecture and lab materials, tests and exams, and other materials, are also protected by copyright and remain the intellectual property of their respective author(s).

Students registered in this course may take notes and make copies of course materials for their own educational use only. Students are not permitted to reproduce or distribute lecture notes and other course materials publicly for commercial or non-commercial purposes without express written consent from the copyright holder(s).

Academic Integrity

Students should be aware of their obligations with regards to academic integrity. Please review the information about academic integrity provided at the Office of Student Affairs website:

carleton.ca/studentaffairs/academic-integrity

This site also contains a link to the complete Academic Integrity Policy that was approved by the University's Senate.

Academic Accommodations

You may need special arrangements to meet your academic obligations during the term. To request academic accommodation, the processes are as follows:

Pregnancy

Email your instructor with any requests for academic accommodation during the first two weeks of term, or as soon as possible after the need for accommodation is known to exist. For more details, read the *Student guide to academic accommodation*, which is available from the Equity Services website: carleton.ca/equity

Religious Obligations

Email your instructor with any requests for academic accommodation during the first two weeks of term, or as soon as possible after the need for accommodation is known to exist, but in no case later than the second-last week of classes. For more details, read the *Student guide to academic accommodation*, which is available from the Equity Services website: carleton.ca/equity

Academic Accommodations for Students with Disabilities

The Paul Menton Centre for Students with Disabilities (PMC) (website: carleton.ca/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 your instructor 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. **Requests made within two weeks will be reviewed on a case-by-case basis.** After requesting accommodation from PMC, meet with your instructor to ensure accommodation arrangements are made. Please consult the PMC website (carleton.ca/pmc/students/dates-and-deadlines) for the deadline to request accommodations for the formally-scheduled exam.

Topics

Most of the code examples will be written in C, but for comparison purposes code written in other languages (e.g., Python or Go) may be presented. C will be used in the labs, but from time to time, we may post exercises (to be completed on your own time) that introduce you to other programming languages.

- Fundamental elements of imperative programming languages: types, variables, expressions, control flow: conditional statements, iteration (loops), functions (subroutines/procedures).
    - Introduction to/review of C as an imperative programming language: types, variables, expressions, `if`, `while`, `for` and `do-while` statements, function definitions vs. function declarations.
- 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 and lists.

    - C arrays. Brief introduction to C character strings.

- Motivation for modular programming. Modules: interface vs. implementation.
    - Modules in C: header (`.h`) and implementation (`.c`) files. The C preprocessor. Compiling and linking C programs comprised of several modules. Brief overview of the standard C library.
- Pointers. Depicting pointers in memory diagrams.
    - C pointers. The `&` and `*` operators. Passing pointers to local variables as function arguments. Drawing memory diagrams to explain how pointers are passed as function arguments.
    - C arrays and pointers. Drawing memory diagrams to explain how pointers to arrays are passed as function arguments.
- Structuring data: structures.
    - C `struct`s. Structures vs. pointers to structures as function arguments. The `->` operator. Memory diagrams.
- Introduction to dynamically-allocated memory and the heap.
    - Heap management in C: `malloc` and `free`. Drawing memory diagrams to explain how parameters and local variables in activation frames can point to memory blocks allocated on the heap. Memory leaks. Dynamically allocated `struct`s.
- Dynamically-allocated arrays, dynamic arrays.
    - Case study: C implementation of a list collection using a dynamic array.
- Structuring data: linked lists. Drawing memory diagrams to understand the fundamental operations on singly-linked lists.
    - Implementing linked lists in C, using `struct`s and pointers.
- Dynamically-allocated arrays, dynamic arrays.
    - Case study: C implementation of a list collection using a dynamic array.
- Abstract collections: stacks and queues. Comparison of different designs, based on arrays and linked lists.
    - Implementing stacks and queues as C modules.
- Introduction to recursion.

Edited: April 12, 2018