CSC 230, C and Software Tools, Section 651
Summer, 2025
Course Syllabus

Instructor:
Fardin Saad
Email: fsaad@ncsu.edu
Office Hours: Tue/Wed 03:00 – 04:00 pm
Zoom: Link
Note: We will follow Dr. David Sturgill's video lectures from the Spring 2025 semester.

Teaching Assistants:
Uchswas Paul (upaul@ncsu.edu)
Office hours: Wed/Thur 02:00 – 03:00 pm
Zoom: Link
Rawshan Mowri (rmowri@ncsu.edu)
Office hours: Mon/Fri 04:00 – 05:00 pm
Zoom: Link

Text:
C Programming, A Modern Approach, 2nd Edition, King

Course Description

In this course, students will develop skills in several important areas. First, we'll get some experience working in C, a language that lets us think like procedural rather than an object-oriented developers. Also, being a fairly low-level language, C lets us see and control more of what's going on in the hardware. This can help us think about using the hardware more effectively, whether we're actually programming in C or in a higher-level language. While we learn C, we'll also learn about tools and techniques that help us build, manage, debug and analyze software projects.

The computer science department captures the essential content of all undergraduate courses in the form of a list of course objectives. This helps to make sure that the most important material gets emphasized even if different instructors lead the course from semester to semester.

Course Objectives

By the end of the course, students should be able to do the following.

- Compilation: implement C programs using the C standard library, with separately-compiled modules; explain the steps of compilation; identify and fix errors that happen during compilation and execution.
- Language: write, debug, and modify C programs using data types, control structures, operators, library utilities, and variables, including scope in a single function, across multiple functions, and across multiple modules.
- Assembly language: describe and explain a small subset of assembly language sufficient to illustrate implementation of C features like the switch and goto statements, accessing parts of the stack frame and accessing fields in a struct.
- Numbers: add and subtract unsigned and signed, two's complement binary integers and convert among standard types including bases 2, 10, and 16; describe 16-bit and 32-bit IEEE floating representation and its consequences for rounding error, and convert between these formats and decimal.

- Memory and Representation: use functions and basic data structures involving arrays, structs, pointers, and function pointers; allocate and deallocate memory in C programs while avoiding memory leaks and dangling pointers.
- Tools: utilize software development tools to implement, test, build, and trace C programs including, build automation, version control, static analysis, and dynamic analysis tools.
- Security: describe and demonstrate how to avoid common programming errors that lead to security vulnerabilities, such as buffer overflows and injection attacks; describe security properties provided by cryptographic primitives (e.g., symmetric cryptography, asymmetric cryptography, and cryptographic hash functions).

Grading

Your final grade in this course will reflect your score on two preliminary exams, a comprehensive final exam, small, programming exercises, larger programming projects and online quizzes. These will be combined with the following weights:

Online Quizzes	8
Programming Exercises	8
Projects	40
First In-Class Exam	12
Second In-Class Exam	12
Final Exam	20

Grades on quizzes, exercises, projects and exams will be reported in the Moodle gradebook. Moodle will automatically compute a semester average, but it probably won't be perfect since it will be missing some grades until the end of the semester. Also, Moodle doesn't know our policy for dropping quiz and exercise grades (generally, your average on these two items will be better than the one computed by Moodle). If you need to check your exact average, you'll want to compute it yourself.

After your final average is computed, your letter grade is determined based on the following table. If your average is at least the value on the left, you are guaranteed a grade that's at least as good as the one on the right. For example, if you end up with an average of 86, you will get at least a B. You may even get a B+ if, for example, your performance shows a trend of improvement through the semester or you've taken advantage of opportunities for extra credit.

Minimum Score	Letter Grade
97	A+
93	A
90	A-
87	B+
83	B
80	B-
77	C+
73	C
70	C-
67	D+
63	D
60	D-

Students auditing this class are required to take all the exams and to earn an exam average of 60 or higher. Students enrolled for credit only must earn a course average of C- or better across all components of the course (exams, projects, quizzes and exercises) in order to receive a grade of S.

Evaluation

Exams

We have three exams in this course, two preliminary exams and a final exam. The windows for taking the first exam is June 09 - 10 and the second is July 07 - 08. The window for the final exam is July 28 - 29. Material in this course builds from basic language elements to larger ideas and constructs. This affects the exams. The second exam is intended focus on material covered since the first exam, but a good understanding of all previous material is necessary to do well on this exam. The final exam is intended to be comprehensive. All exams are closed book, but students are permitted to make one 3×5 note card for each exam. You may use both sides of your card, but your card must be hand written by you.

Quizzes and Exercises

In this class, you get to earn some points toward your final grade through a number of online quizzes. Quizzes will be taken online through the course Moodle page.

Short programming exercises will give you a chance to write or fill in missing parts of short programs. They will generally be due the on Friday evening after the relevant lecture material is presented. This will include an exercise deadline of July 25, for the last exercise(s).

Hopefully, the quizzes will give you a good way to see how well you understand the material after each lecture. The exercises should give you an opportunity to practice the programming language concepts as we learn them, instead of just trying to apply concepts when you're working on a project.

I understand that students sometimes have to miss class, or forget about an exercise. You'll get to drop your three lowest quiz grades and your three lowest exercise grades. You'll also get to drop 5 percent of the remaining grade from each of these categories. This is our mechanism for handling any missed quizzes and exercises; it will let you miss a few and do poorly on a few more without any penalty to your final grade.

What does it mean to drop 5 percent of your quiz grade? After dropping the three lowest quiz grades, I drop another 5 percent by adding up all the points available on the quizzes that remain. I reduce this by 5 percent and call it your adjusted maximum, m. Then, I add up all the points you earned on quizzes, clamp that to $[0, m]$ range and divide by m (so, your quiz average won't exceed 100 percent, even if your total exceeds the adjusted maximum). Your exercise average is computed the same way.

Projects

In addition to the frequent, small programming exercises, you'll get to complete five small projects. These will give you a chance to use what you're learning to solve larger, more interesting problems. As these are assigned, they will be posted to the course homepage. Electronic submission of projects will normally be due at 11:59 pm on their due date, with the last one due July 23.

If you have a conflict on the due date, just plan to complete and submit your work early. Since project submission is electronic, you can even submit your work while you're away if you have to travel. If a documented emergency (e.g., hospitalization) prevents an assignment from being submitted, we can agree on how to handle the missed grade, possibly by dropping that assignment from the student's average.

Projects may vary in points. For example, an easy one may be worth 70 points and a more difficult one may be worth 120. To compute your project average, just add up all the points you earned and divide by the total number of points available. So, projects that are worth more points will carry more weight in the average.

Reading Assignments

Regular reading assignments are included in the schedule of topics on the course homepage. The reading will support much of the class discussion, and keeping up with the readings can help you to do well on exercises and projects.

Programming Guidelines

You will get a chance to do some C programming on exercises and projects. Style requirements for the exercises are not strict; unless it's specifically part of the problem you're solving, you can generally format your code and use any variable names you'd like.

Style requirements for the programming projects are strict, very strict. The course homepage includes a document describing the style guidelines, along with some tips about common problems students have. Why are the requirements so strict? In the workplace, it's common to have a set of style guidelines that all developers are expected to follow. In this class, we try to help students get used to this idea.

Programming assignments are expected to compile and execute on what we're calling our *common platform*. This is an installation of Linux that's available to all students. The course homepage includes a description of this platform, along with some help with developing and testing your code on this platform.

Course Policies

Late Submission

If you miss the submission deadline for a project, you can submit it up to 48 hours late for up to a 20 percent penalty. The late penalty is graduated; submitting within 24 hours of the deadline will only cost 10 percent.

If you make a submission before the due date, then, after the due date, you realize that you've made a mistake, you can still make a late submission if it's within 48 hours of the deadline. We will grade the last submission you make before the late deadline. Of course, you'll need to decide whether it's better to lose points for a mistake on an on-time submission or to lose up to 20 percent for a perfect but late submission. In general, you'll earn more points with a correct-but-late submission than you will with a broken-but-on-time submission.

Project Extension Requests

If an emergency prevents you from completing a project by the deadline, you can request an extension. To do this, you'll need to send an email to your instructor. Here's what to do and include in your email.

- Be sure to submit everything you have done to Moodle (for project 1) or push it to git (for projects 2 - 5). This will help to show that you've made a start on the project and that it's the unexpected emergency that's preventing you from completing it in time.
- Be sure to put the word "extension" in the subject line of your email.
- In your email, give a brief explanation of what's preventing you from completing the project.
- If possible, go ahead and say how much additional time you'll need to complete the assignment. This is something that can be worked out or adjusted later, but it might save some time to go ahead and say how much of an extension you are asking for.
- Include relevant documentation. Typically, this might be be a doctor's note with a date. It's OK if the documentation isn't available at the time of the request; you can email it later as you and your instructor agree on an extended deadline.

A request email for a project extension might look something like the following:

```
To: peggy_professor@ncsu.edu
From: sylvester_student@ncsu.edu
Subject: Project 1 Extension

I'd like to requestion an extension for project 1.  I broke my arm on
Monday afternoon, and it's slowing down my typing quite a bit.  I
think my project is about 70 percent finished already, and I've
```

```
submitted the work I've already done on Moodle.

I should be able to finish the project if I can have 48 more
hours to work on it.  I don't have a doctor's note yet, but I should
be able to send you one on Thursday.

Thanks.
```

Make-Up Exam Requests

Exams days are scheduled from the start of the semester, and you should make sure you don't schedule an event or a meeting that conflicts with one of the exams. In the distance section of the class, you have a window of several days to schedule your exam, so you should be able to avoid conflicts with other meetings or events. If and only if, an unscheduled emergency prevents you from taking one of the exams, we can plan for a make-up. To request a make-up exam, email your instructor with the following:

- Be sure to put the word "exam" in the subject line of your email.
- In your email, give a brief explanation of what's preventing you from taking the exam as scheduled.
- If possible, say something about when you would be able to take a make-up exam. Of course, this will depend on the type of emergency, but it might help speed up the process a little bit.
- Include relevant documentation. Typically, this might be be a doctor's note with a date. It's OK if the documentation isn't available at the time of the request; you can email it later as you and your instructor plan for the make-up exam.

Exercise and Quiz Extensions

We don't normally grant extension on the daily exercises. These are short and submitted electronically, they're worth only a small part of your grade, and the drop policy lets you miss a few deadlines without affecting your exercise grade. If a protracted illness prevents you from meeting several of the exercise deadlines, you and your instructor can consider options for dropping or making up part of your exercise grade.

Regrade Requests

On exams and programming projects, the teaching staff establishes guidelines for grading, and grading responsibility is shared with within the staff (Instructor and TA/Graders).

- **Exam:** This semester, we're using gradescope to help manage exam grading. For these, we'll use the regrade system built into gradescope. Students have one week from when the graded exam is returned to begin an appeal.
- **Projects/Exercises:** If you believe an error has been made in grading a project or exercise, write up a short description of your case and send it to the TA/Grader responsible for the project/exercise (email is fine).

Class Membership and Participation

Class Communication

We will use a shared **Piazza** platform with the on-campus sections to facilitate out-of-class discussions. A link to the Piazza page is available on the course homepage. Students are encouraged to post questions and support one another by responding to peers' queries.

To keep online section concerns distinct from those of the on-campus sections, we will also use **Moodle**. The *Announcements* section in Moodle will be used by the teaching staff for important updates (e.g., grade postings, project modifications). Students may post questions in the *Student Message Board* section.

All posts to the message board must follow three requirements:

- The post must be courteous to and respectful of students, staff, and the University community.
- Before you post a new question, have a quick look to see if the same question has already been asked (and maybe answered). Reducing duplication in the discussion board will make it more useful to everyone.
- Students must consider the academic integrity expectations and the difference between answering a question and providing a solution. It's OK to help out by explaining a problem you had, posting sample input and output, or pointing out a tricky special case. It's not OK to publicly post your code (even broken code or just a line or two) when you're asking for help, and it's not OK to post a working solution (even in part) when you're trying to help someone else out.

After the semester starts up, we'll depend on piazza and the Announcement section in Moodle for class announcements, so make sure you're registered with our piazza site and have subscribed to the Announcement section for receiving the notifications.

Dissemination of Information

The course website in Moodle is the primary distribution medium for this course. Assignments, lecture slides, many examples, study guides and other materials will be available from this site.

Office Hours

The teaching staff will hold online office hours via zoom, which students are welcome to attend virtually. You can find the zoom link at the beginning of this document or in Moodle.

Academic Integrity and Attendance

You are expected to complete your own work independently and refrain from assisting others in a way that would allow them to bypass doing their own work. Academic integrity is taken seriously and any violations will be addressed in accordance with university policy.

As this is a Distance Education course, real-time attendance during lecture uploads is not required. However, to be marked present, you must complete the quiz that accompanies each lecture video once it is posted.

Academic integrity can sometimes be a problem, so this document gives it some extra attention to help people stay out of trouble. You may only work on an assignment with another student if the assignment explicitly states that this is permitted. In completing your assignments, you can talk to the instructor or the teaching assistants to get specific help. You can use materials posted on the Moodle course homepage and examples from your book. You can talk to anyone about some general techniques for solving a problem, but you should never share copies of the source code or other parts of your work with someone who is not on the teaching staff for this course.

Assignments in this course are intended to help you develop your skills in problem solving, design, coding and communication. Any material you submit must be solely your intellectual work. You should never turn in work that contains material copied from some other source, even if you made modifications to the work before turning it in. You should never turn in work that is, even partially, from generated material.

For example, you are not allowed to use any artificial intelligence (AI) tools, such as chatbots, text generators, paraphrasers, summarizers, or solvers, to complete any part of your assignments. Use of these tools would be considered an academic integrity violation and will be dealt with according to the university academic integrity policy.

It is your responsibility to understand the origins of the work you turned in. If your development environment includes features to generate code or text automatically for you, you may have to disable or

decline to use these features in order to make sure your submissions are entirely your own work. It's OK if your editor helps to indent code for you, matches or inserts closing parentheses and brackets, performs spell checking or auto-completes identifier names, but you should make sure it doesn't create larger parts of your solution for you.

Students are expected to maintain high standards of academic integrity and honesty. Normally, a case of cheating will result in a grade of zero for an assignment. A major offense, including any violation on a test, could result in failure of the course.

Suspected violations of academic integrity will be reported to the Office of Student Conduct. This benefits the student in that it provides an opportunity for independent review of the evidence. It also benefits the university by maintaining a centralized record of violations.

Examples of Cheating (this list is NOT exhaustive):

- to give any student access to any of your work which you have completed for individual class assignments.
- It is cheating AND plagiarism to use another person's work and claim it as your own. You are expected to complete all assignments on your own, unless otherwise specified in the assignment.
- to interfere with another student's use of computing resources or to circumvent system security.
- to email, ftp, post on the Internet, bulletin boards, message boards, etc. your work for others to obtain. Do NOT use sites that allow you to anonymously post code. Those sites are searchable, and others may find your code (like the teaching staff).
- to ask or pay another person or persons to complete an assignment for you.
- It is cheating AND plagiarism to decompile any compiled code and use the decompiled source code as your own. You may also break the law by decompiling code.
- It is cheating AND plagiarism to use code that you find online.
- to give another student access to your account (NC State account or others that you use for university work) or to give them your account password.
- for you and another student to work collaboratively on an assignment, unless otherwise specified by the assignment.
- to circumvent the intention of the assignment and/or the automated grading system (e.g., by hardcoding test case solutions).
- to use generative AI tools (e.g., ChatGPT, Copilot, etc.) to write or complete code unless explicitly permitted by the assignment instructions.
- to submit code or solutions reused from prior semesters or shared by former students, even if slightly modified.
- to misrepresent the authorship of group work by submitting contributions you did not make, or by failing to properly acknowledge your collaborators (when collaboration is allowed).

Examples of NOT Cheating (this list is NOT exhaustive):

- Using the code from the class website (with citations in the comments).
- Using code from other programs YOU wrote.
- Help from TAs or instructor (with citations in the comments).
- Using code from the textbook or textbook website (with citations in the comments).

Example Citations

```
/* (In file or function level comments)
 * I received help from the TA, Martha Washington on DATE during her office hours.
 * We discussed X.
 */

/*
 * The code for this method is based on Exercise Y that I completed on date Z.
 */
```

Protecting Yourself

- Do not leave papers lying around your workstation.
- Do not dispose of important papers in the lab recycling bins and trash cans until after the assignment is graded.
- Do not give out your password.
- Do not leave your workstation unattended or forget to log yourself out.
- Do not leave your laptop unattended.
- Do not give other students access to any of your workspace or email them any code.
- Do not give other students access to your course materials or your personal computer.
- Do not email, ftp, or post your code on the Internet, message boards, etc.
- Keep all copies of final an intermediate work until after the assignment is graded.
- Keep all graded assignments until after you receive the final grade for the course.
- Do not discuss implementation details of the assignment with your peers.

Standard Syllabus Statements

All NCSU syllabi are required to have standard paragraphs describing course policies in particular areas. These apply to our course. They're probably the same as statements you've seen in other course syllabi at NCSU.

Statement for students with disabilities

Reasonable accommodations will be made for students with verifiable disabilities. In order to take advantage of available accommodations, students must register with the Disability Resource Office at Holmes Hall, Suite 304, 2751 Cates Avenue, Campus Box 7509, 919-515-7653. For more information on NC State's policy on working with students with disabilities, please see the Academic Accommodations for Students with Disabilities Regulation (NCSU REG 02.20.01).

Privacy and Interactions

Students may be required to disclose personally identifiable information to other students in the course, via digital tools, such as email or web-postings, where relevant to the course. Examples include online discussions of class topics, and posting of student coursework. All students are expected to respect the privacy of each other by not sharing or using such information outside the course.

Semester Schedule

The following table gives the planned schedule of lecture topics, readings, exams and other important events for the semester. We'll try to stick to this schedule, especially for the planned exams. It's based on the schedule for one of the on-campus sections, but, as a student in the online class, you have some flexibility in when you view the lectures. This will give you some guidelines for when you should view them if you want to be prepared for the exams and assignments. If weather requires the on-campus sections to miss a lecture or two, or if we slip the deadline on a programming project we may need to adjust this schedule a little bit.

Date	Events
05/14/2025	Lecture 01, Getting Started Reading: Chapter 1 (Introducing C) Reading: Chapter 2 (C Fundamentals)
05/16/2025	Lecture 02, I/O and Compilation Reading: Chapter 3 (Formatted I/O)
05/19/2025	Lecture 03, Revision Control and GDB
05/21/2025	Lecture 04, Variables and Types Reading: Chapter 4 (Expressions) Reading: Chapter 5 (Selection Statements) Reading: Chapter 7 (Basic Types)
05/23/2025	Lecture 05, Program Organization Reading: Chapter 6 (Loops) Reading: Chapter 9 (Functions) Reading: Chapter 10 (Program Organization) Reading: Chapter 18 (Declarations)
05/25/2025	**Project 1 Due**
05/26/2025	*Memorial Day (No Classes)*
05/28/2025	Lecture 06, Arrays and Strings Reading: Chapter 8 (Arrays) Reading: Chapter 13 (Strings)
05/30/2025	Lecture 07, Multi-Dimensional Arrays and Build Automation
06/02/2025	Lecture 08, Pointers part 1 Reading: Chapter 11 (Pointers)
06/04/2025	Lecture 09, Integer Representation
06/06/2025	Lecture 10, File I/O Reading: Sections 22.1 - 22.3 (Streams and Formatted I/O)
06/09/2025 - 06/10/2025	**Exam 1** (Lectures 01 - 09)
06/11/2025	Lecture 11, Pointers part 2 Reading: Chapter 12 (Pointers and Arrays)
06/13/2025	Lecture 12, Dynamic memory allocation Reading: Sections 17.1 - 17.4 (Dynamic Allocation)
06/15/2025	**Project 2 Due**
06/16/2025	Lecture 13, Pointers part 3 Reading: Sections 17.6, 17.7
06/18/2025	Lecture 14, Structs Reading: Chapter 16 (Structures, Unions and Enumerations)
06/20/2025	Lecture 15, Debugging Reading: Section 17.5
06/23/2025	Lecture 16, Data structures
06/25/2025	Lecture 17, Bitwise operators Reading: Section 20.1 (Bitwise Operators) Reading: Section 20.2 (Bit-Fields in Structures)

06/27/2025	Lecture 18, Math and Floating Point Reading: Chapter 21 (The Standard Library) Reading: Sections 22.4 - 22.8 (Character/Block/String I/O)
06/29/2025	**Project 3 Due**
06/30/2025	Lecture 19, The preprocessor Reading: Chapter 14 (The Preprocessor)
07/02/2025	Lecture 20, Program Organization part 2 Reading: Chapter 15 Reading: Chapter 19
07/04/2025	*July 4th (No Classes)*
07/07/2025 - 07/08/2025	**Exam 2** (Lectures 10 - 18)
07/09/2025	Lecture 21, Abstraction
07/11/2025	Lecture 22, Assembly Language and Implementation Part 1
07/13/2025	**Project 4 Due**
07/14/2025	Lecture 23, Assembly Language and Implementation part 2
07/16/2025	Lecture 24, Assembly and The rest of C Reading: Chapter 23 (Library Support) Reading: Section 26.1 (Vararg Support) Reading: Section 26.2 (General Utilities)
07/18/2025	Lecture 25, Security
07/21/2025	Lecture 26, Performance
07/23/2025	**Project 5 Due**
07/25/2025	**Reading/Discussion**
07/28/2025 07/29/2025	**Final Exam**

Course Materials Acknowledgment

This course features materials provided by other instructors at NC State. Thanks to Dr. David Sturgill, Dr. Sarah Heckman, Barry Peddycord and Dr. Douglas Reeves for generously providing their materials and assistance and helping to make this course possible.