


# CSCI-243: The Mechanics of Programming

## Syllabus: Fall 2023 (2231)



---

[Syllabus](#)   [Schedule](#)   [Resources](#)   [Instructors](#)   [TA](#)   [RIT Academic Calendar](#)    \*

---

*Last revised:* Thu Sep 21 14:29:10 EDT 2023

---

## 1. Catalog Description

This course introduces you to the details of program structure and the mechanics of execution as well as supportive operating system features. Security and performance issues in program design will be discussed. The program translation process will be examined. Programming assignments will be required.

---

## 2. Course Outcomes

Upon successful completion of this course, you will be able to:

- Design and implement solutions to problems of moderate complexity using an imperative programming language.  
*Evaluation:* Programming assignments, exams.
  - Explain how program design can affect the performance and security of executing software.  
*Evaluation:* Programming assignments, exams.
  - Describe aspects of computer operating systems that support operations and activities of application-level programs.  
*Evaluation:* Exams.
- 

## 3. Contact Information

See the [Instructors](#) link.

---

## 4. Course Prerequisites

There are three possible prerequisites for enrolling in CSCI 243 (Mechanics of Programming). You must achieve a minimum grade of C- in one of these courses to satisfy the prerequisites:

- Computer Science II (CSCI-142), or
- Computer Science for AP Students (CSCI-242) or
- Computer Science for Transfer Students (CSCI-140).

You should be familiar with the creation and use of standard data structures (linked lists, stacks, queues, trees, etc.), recursion, iteration, and object-oriented concepts.

---

## 5. Textbooks and Course Materials

There is no required textbook for this course. For those students who prefer to have a textbook, we have this recommendation: *Programming in C, A Complete Introduction to the C Programming Language*, fourth edition, by Stephen G. Kochan; Pearson Education, 2015. Readings from this text are indicated as [Kochan4, 2015] in the schedule on the [course account home page](#). The older (third) edition of this text is also acceptable.

The text book is a valuable reference and source of quality materials. Other sources that you may find online are not necessarily as reliable or consistent as a published text; we have listed several of the better ones on our [course resources](#) page.

This course is not entirely about programming in C, which means that there are topics which do not have corresponding book content. Other course material will be published through handouts or online through the [course account home page](#). You should also expect to take notes during the lectures to capture important terms and concepts for later review and study.

---

## 6. Assignments

Course assignments include exams, homeworks, and projects. Your instructor may also elect to give additional *in class* assignments.

### 6.1. Exams

There will be three exams: two mid-term exams, and a common, comprehensive final exam.

You are expected to take exams during their scheduled time periods. In general, there will be no make-up exams. However, we recognize that situations could arise that would prevent you from taking an exam (severe illness, accidents, etc.). Should such an emergency occur, you must inform your instructor prior to the exam. You may email your instructor, call them, or leave a message with the staff in the Computer Science Department office (GOL-3000, telephone 475-2995 or 475-6179). You must make arrangements for a makeup exam when you return.

**Please note that oversleeping, cars that don't start, and other excuses of this kind are not acceptable. It is your responsibility to get to class on time for exams. If you miss an exam and did not make prior arrangements for a makeup, you will receive a zero for it.**

### 6.2. Programming Assignments

Programming assignments will be of two types: *homeworks* and *projects*. Homeworks are programming assignments that are designed to give you experience with concepts, tools and techniques you will need to apply in larger assignments. Projects are those larger assignments; they build upon your experience with homeworks.

There are no formal lab sessions for this course. You will develop and submit solutions to all assignments independently, outside of class.

All programming assignments for this course are to be submitted using the computing facilities provided by the CS Department. These systems are 64-bit Intel-family computers which are configured to be as close to identical (in terms of available software) as possible; all are running the Ubuntu<sup>®</sup> 20.04 operating system, and have the GNU C compiler gcc installed (currently, GCC version 9.4.0). We will be using the 1999 version of the C standard produced by the International Organization for Standardization (ISO), commonly referred to as the *C99 language standard*.

## 6.3. Grading Evaluation and Feedback

For homework assignments, you should expect to receive your grade and feedback comments within two weeks of the late submission deadline for that assignment. Project assignments generally take longer to grade, so you should expect to receive those within three weeks of the late submission deadline. You should expect exam grades to be returned within two weeks of the exam date.

All programming assignment grades are returned via email from the course grader account to your RIT email address ([loginid@rit.edu](mailto:loginid@rit.edu)).

## 6.4. Programming Assignment Submission and Testing

The submission system used by the CS Department is known as `try`. This tool will automatically compile, link, and test your program submissions, recording the results for our examination when your submissions are graded. The [course resources](#) page provides information on its use in the form of two documents:

- The [try summary](#) is a quick list of suggestion about using `try` - things to do, things to watch out for, and (most importantly) things to *not* do.
- The [Student Guide to try](#) is a more detailed look at `try`.

**Submissions must pass through the `try` system for compilation, linking, and testing in order to be accepted. No emailed assignment solutions will be accepted.**

In all cases, the minimum acceptance test for programming assignment submissions is that the submission must compile and link cleanly when submitted through `try`. This means there must be no fatal compilation or linking errors, and there must be an executable program image available for testing.

For most assignments, `try` will also be performing a series of runtime tests on your submission. Some of these may be identified as *mandatory tests* of your submission; in this case, those tests must be completed satisfactorily, which generally means the output must be correct when compared against the assignment specification. (When an assignment has mandatory tests, they are typically simple tests of the basic functionality of the code.)

**Should your submission fail to compile and link cleanly, or if it fails any mandatory tests, it will be rejected by `try` and will not be accepted for grading.**

When testing assignments, `try` will typically be configured to enforce one or more of the following limits on the executing program:

- *Runtime limits* guard against programs going into infinite loops
- *Output size limits* are protection against programs creating massive amounts of output text
- *Memory size limits* are protection against programs which allocate so much memory at runtime that they cause problems for the entire system.

In all of these cases, the limit will be generous enough that only programs with serious logic errors or exceptionally inefficient algorithms will be caught by them. If you believe your program has incorrectly been caught by one of these, see your instructor.

It is possible that, in spite of the protections described above, a submission may “hang” when run under `try`. (Generally, the symptom of this is that a test being run appears to be taking an excessive amount of time, on the order of many minutes.) A submission that “hangs” *will not make it past try*, and you may need assistance to resolve this. ***Do not wait until the last minute to submit!***

All assignment submissions will be tested using machines configured the same way as the machines in the CS labs (ICL1, ICL2, and ICL3) and the CS compute servers (`glados.cs.rit.edu` and `queeg.cs.rit.edu`). These machines are all 64-bit architectures with Intel CPUs.

**Important Note:** The CS systems can get busy near the submission deadlines. This is **not** an excuse for not meeting the deadline! It is your responsibility to make sure you have enough time to submit and allow all the tests run to completion.

You are encouraged **not** to work locally on your own computers. (Using `ssh` from your computer to a CS machine is fine; here, we are talking about using native editors and compilers on your home system.) Should you choose to do so anyway, consider the following caveats:

- While C source code is generally portable between computer systems, there are differences in the way the C function libraries are implemented, differences in the support provided to executing programs by different operating systems, and even differences in the way C header files are structured and in the contents of those header files. In addition, there are parts of the C language specification that specifically define program behavior as *implementation-dependent*. This means that some C compilers may generate code that executes in different ways than other C compilers and can lead to programs producing different results. This leads to runtime incompatibilities between computer systems.
- If you use a Linux<sup>®</sup> system and compiler that are different from the ones installed on the CS systems, make sure the computer is configured with a 64-bit operating system version of Linux and the 64-bit version of the GNU C compiler installed on it. On any CS Linux machine, you can use the following commands to determine the characteristics of that system:
  - `lsb_release -a` will report the OS version
  - `uname -a` will tell you the machine architecture and the OS kernel version
  - `gcc --version` will report the compiler version
- If you use a Macintosh system, the standard compiler is named `gcc` but is a different compiler from that on Linux systems. This means that your results will likely be different.
- If you use a Windows system, be aware that Microsoft's C compilers often implement non-standard versions of the language. This means that standard C features may not be supported, and things that are not part of the C99 standard may be accepted.
- The `try` system works only on CS department Linux machines. You must have your code on the CS servers and be logged into a CS department Linux machine to be able to submit. Before you submit work written on a host platform not in the CS domain, To submit work you have done on a non-CS system, you will have to do all these steps:
  - Upload the code to the CS systems;
  - Compile and link the code (you may find differences then);
  - Retest the code (and find more differences); and finally,
  - Run the `try` submission command.

That takes significant extra time before you can submit.

**Programs that “work on my computer at home” but do *not* work on the CS systems do not work, and will be graded accordingly.**

## 6.5. Assignment Submissions and Late Policy

Each programming assignment submission is due electronically by 11:59:59 pm (23:59:59 for those who prefer 24-hour time) on its published due date. Every assignment will have a due date published on the assignment

writeup. In general, assignments will be due as follows:

- *Homework assignments* are typically due approximately seven calendar days after the assignment was published; some more involved assignments may have later due dates.
- *Project assignments*, which are larger tasks than homework assignments, typically have due dates that are three to five weeks after the date on which they were published.

Most programming assignments can be submitted up to 24 hours following their published due dates under our *late submission* policy. Submissions which are made no later than the assigned due date are considered "on time"; those submitted after that time but no later than the late submission deadline are considered "late", and are subject to a late penalty.

The late penalty for programming assignments applies a "ceiling" to the earned grade for the assignment. The submission is graded normally; if it was submitted late, the grade is limited to **80% of the original maximum score**. A score that would have been higher than this ceiling is capped at 80%; a score that is lower than the ceiling is not affected, and thus no penalty is assessed for that late submission.

**The latest dated submission for an assignment is the one that will be graded. If there are both on-time and late submissions for an assignment, this means the late submission will be the one that is graded.**

## 6.6. Coding Style and Revision Management

Please follow a reasonable programming style. **Your code must be neat, clear, documented and above all consistent.**

See the [course resources page](#) and your instructor for details on formatting style and documentation standards. Below is a brief list of things expected regarding code style.

1. reasonable length function bodies ( $\leq 1$  page, 25-50 lines);
2. reasonable length lines of code ( $< 80$  characters);
3. program file header comment block describing the file content and containing *your full name* (first and last names; 'Joe' is not a complete name, nor is Joe's CS account name);
4. publishable, function header documentation describing the purpose, parameters, return values, etc. of each function;
5. clear, consistent indentation (*no mixing TABs and spaces*);
6. internal documentation of complex sections of code; and
7. evidence of *version control*, either inside or outside the file.

Most assignments will have a grade component for use of version control (the first assignment will not include this). The current standard version control tool is git, and you are expected to use git in later assignments.

---

## 7. Grading Scale and Weights

Your grade as a percentage translates to letter grades using the following scale:

Letter	Percentage Range
A	92% or above
A-	at least 89% but under 92%
B+	at least 85% but under 89%
B	at least 82% but under 85%
B-	at least 79% but under 82%

C+	at least 75% but under 79%
C	at least 72% but under 75%
C-	at least 69% but under 72%
D	at least 60% but under 69%
F	under 60%

Gradable assignments are separated into two classes: programming assignments, and exams. The table below shows the weighting of these components.

Component	Elements	Weight	Notes
Programming Assignments (50%)			
	Homeworks	25%	Do all work independently, outside class. All homeworks have an equal grade weight. The lowest homework grade will be automatically dropped (unsubmitted assignments will be considered to have grades of zero for this purpose).
	Projects	25%	Do all work independently, outside class. Project 1 is 10% of the course grade, and project 2 is 15% of the course grade.
Tests (50%)			
	Midterm Exam 1	15%	The exam is a full lecture period written test.
	Midterm Exam 2	15%	The exam is a full lecture period written test.
	Final Exam	20%	The final exam is a comprehensive, written exam, given during the final exam period.

Programming assignments will be randomly "cheat-checked" through the use of code analysis tools.

## 7.1. The Course Grade Limit Rule

In this course, a *grade limit rule* is used to deal with disparities between the two major grading components (programming assignments, and exams). Specifically, your final course grade may be *at most* 10 percentage points higher than the lower of your two component averages.

As a matter of practice, this limit comes into play only when the difference between your Programming Assignment and Test averages is more than 20%. It has been the experience of the department that a student whose grade difference is this great is having “more difficulty than meets the eye” with regard to understanding and mastering the material.

Here is an example. Consider a student who receives a 68% average on the Tests component and a 96% average on the Programming Assignments component. In this case, the student's course grade would be limited to 78% (a C+), which is 10% above the Tests component grade. Without the grade limit rule, the final weighted grade would have been  $68 * .5 + 96 * .5 = 82\%$  (a B).

## 7.2. Grade Appeals

All requests to regrade any assignment must be made within one week of the date on which the assignment was returned. After that time, the grade becomes permanent. (As an example, you may not ask for a regrade of a homework score returned to you in week 4 when it is week 7.) Requests must be made through email to your instructor.

## 8. Wellness

Success in this course depends heavily on your personal health and well-being. Stress is an expected part of the college experience, and unexpected social and personal setbacks can compound this. We instructors encourage you to view these challenges as part of the path to success.

The academic demands of this course and other classes can be understandably difficult. It is normal to feel anxious about your academic ability, especially when unexpected life events emerge. The instructors invite you to communicate with us about difficulties you have in this course as soon as possible. Your success is important. We want you to get the additional assistance needed before the challenges become too much.

---


## 9. Getting Help

With greater experience following your first year of CS, there is an expectation that you are better able to “figure it out” when you encounter challenges with the course material.

Nevertheless, everyone gets really stuck sometimes, and you may obtain help from the following sources:

- The [course resources page](#);
- Your lecture instructor during posted office hours or by appointment;
- The course teaching assistant (CTA) for this course. Their schedule of office hours and session times will be published on MyCourses as soon as it is available.

Note: The CTA helps with technical questions about language and programming environments. For example, the teaching assistant might help you use the debugger's commands and explain the nature/structure of its output. The CTA can help explaining dynamic allocation but will not help you redesign and rewrite your project's memory management.

- Unfortunately, the [GCCIS Tutoring Center](#)  on the second floor is NOT available for help with this course. That facility is for first-year courses only, and the CTA is the outside support point for this course.
- 

## 10. Miscellaneous Policy Notes

Please monitor the course web page and MyCourses regularly to stay informed. The instructors reserve the right to make changes to any facet of the course based upon the events of the term. If such a change is made, the change will be advertised/announced in class, via electronic mail, through MyCourses, or in the course web pages (<http://www.cs.rit.edu/~csci243/>).

**Unless otherwise specified in an assignment, all work you submit for grading must be your own, individual production.** You may use “example” code discussed in class or found in the weekly lecture code directories, with proper attribution (e.g., through comments within your code indicating the origin and the modifications you have made to adapt the code to this assignment). Some assignments may suggest online reference material for algorithms or optimizations (etc.); inspiration from this material should be attributed as well.

Due dates for assignments are intended to provide adequate time to complete the assignments, while allowing sufficient remaining time in the term to complete the remaining assignments. Should it become necessary, the instructors reserve the right to change due dates; this, in turn, may require modification of due dates for other assignments during the term, or, in some cases, elimination of some assignments.

System downtime on or near the due date for an assignment is not usually grounds for an extension. An



exception to this is *extended* system downtime (on the order of multiple days, not just hours); if this occurs, instructors may consider modifying a due date, but this is not guaranteed.

---

## 11. Common Course Policies

### 11.1. Disability Services

RIT is committed to providing reasonable accommodations to students with disabilities. If you would like to request accommodations such as special seating or testing modifications due to a disability, please contact the Disability Services Office. It is located in the Student Alumni Union, Room 1150; the Web site is [www.rit.edu/dso](http://www.rit.edu/dso). After you receive accommodation approval, you must see the instructor to work out whatever arrangements are necessary.

### 11.2. Academic Integrity and Academic Dishonesty

Academic dishonesty will be dealt with in accordance with DCS and RIT policies.

[RIT's Academic Honesty Policy](#) defines the basic forms of academic dishonesty (cheating, duplicate submission, and plagiarism) and explains the official RIT policy regarding academic dishonesty.

*Unless otherwise explicitly stated in an assignment writeup*, all Homework and Project assignment submissions must be the result of individual effort, not teamwork. Development of code for all graded work is an individual responsibility.

Submitting individual work written by others or as an unsanctioned team is considered an act of academic dishonesty. In cases where a student is suspected of cheating or copying material, the instructor shall notify the students involved and act in accordance with <http://www.rit.edu/academicaffairs/policiesmanual/d080>.

Although students may discuss assignments with others, all individually submitted writings and code must be created independently by the student and not copied from others or other sources (e.g. web pages). This includes copying from public repository sites such as GitHub. Work copied from GitHub or other, similar sources will be subject to prosecution for breach of academic integrity.

Given the intent of this course, the use of LLMs to generate code or algorithms for assignment solutions is not allowed. (Examples include, but are not limited to, GitHub Copilot, StarCoder, and Google Bard.) Use of an LLM to produce assignment solutions will be subject to prosecution for breach of academic integrity.

Academic integrity includes the understanding that **no student will post their code in publicly accessible storage locations**. You should store work only on private, personal computers or on private, password-protected, unsearchable on-line storage facilities. Hard copies, if created, must be kept secure.

Related to this is the use of external repository sites such as GitHub to hold your coursework. You must not use publicly-accessible repositories on such sites, as this makes your code available to other students; someone copying and submitting your code from such a repository implicates you as the provider of the copied code. Be sure that any such version control repository you use is private.

The [CS Department Selected Policies](#) document explains the official Department policies regarding leaves of absence, incidents of academic dishonesty, and other important-to-know policies.


### 11.3. Other Policies

- Other RIT policies may be found at the provost's governance library, <http://www.rit.edu/academicaffairs/policiesmanual/policies/governance>.



- [RIT's final exam policy](#) 
- The RIT policy on harassment is covered in <http://www.rit.edu/academicaffairs/policiesmanual/c060> .

---

*\* Links marked with the symbol  open in a new browser window or tab.*

*UNIX<sup>®</sup> is a registered trademark of The Open Group.*

*Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.*

*Ubuntu<sup>®</sup> is a registered trademark of Canonical Ltd.*

---

*Revision history:*

2023/09/21: minor formatting tweaks

2023/08/24: original version for 2231

---