Cal Poly Humboldt
Course Syllabus for CS 111 - Section 10
Computer Science Foundations 1
CRN 42028 - Fall 2025

Lecture meets:	Tuesdays and Thursdays, 9:00 am - 10:20 am	NR 101
Lab Section 11 meets:	Fridays, 9:00 am - 10:50 am	BSS 313
Lab Section 13 meets:	Fridays, 3:00 pm - 4:50 pm	Online/Zoom

Instructor:	**Sharon Tuttle - she/her/hers**	
Instructor's e-mail:	st10@humboldt.edu or sharon.tuttle@humboldt.edu or smtuttle@humboldt.edu	(note: these are all ALIASES to the SAME mailbox)
Instructor's office:	BSS 322	(3rd floor, in one of the corners furthest from the elevator)
Student hours:	Tu: 12:30 - 2:30 pm W: 11:30 am - 12:30 pm Th: 12:30 - 2:30 pm or by appointment	(I'll be in BSS 322, but also have a Zoom session running from there if you prefer to use that) (Zoom link: see course Canvas site)
Course public web site:	follow CS 111 link from: https://nrs-projects.humboldt.edu/~st10/ OR follow link from course Canvas site	

Course Description

[from the Cal Poly Humboldt catalog:] Introductory programming covering problem decomposition, control structures, simple data structures, testing, and documentation. Students design and implement a number of programs.

In this course, you will begin to explore the art and science of problem solving using the computer as a tool. Course topics will include problem solving, simple expressions and compound expressions, types of data, syntax and semantics, function application, design, and definition, introduction to software testing, Boolean operations, Boolean functions, conditional expressions, input-process-output, and the basic structures of computing: sequence, conditional, iteration, and procedure.

The problem solving techniques learned in this course lead to a study of object-oriented programming as well as an introduction to pointers and memory management in CS 112 - Computer Science Foundations 2.

Students are expected to come into this course already comfortable with basic computer use. **No prior programming knowledge is assumed or required.** In this particular offering of this course, we will start the semester programming in the Beginning-Student Level (BSL) of the Racket language, and then transition to programming in C++.

Course CO-requisite

Math 101 - College Algebra **or** Math 101I - College Algebra IS **or** Math 102 - Algebra & Elementary Functions, **or** instructor approval.

Notice that the above is a **CO-REQUISITE** for CS 111. That means Math 101 or Math 101I or Math 102 can be taken either before OR *at the same time* as CS 111.

Again, note: **No prior programming knowledge is assumed or required for CS 111.**

Student Learning Outcomes

After successfully completing this course, students should be able to:*

- Design, implement, test, and debug programs that use each of the following fundamental programming constructs: basic computation, standard conditional and iterative structures, and the definition of functions.
- Analyze the behavior of simple programs involving fundamental programming constructs.
- Choose appropriate conditional and iterative constructs for a programming task.
- Apply the techniques of structured (functional) decomposition to break a program into smaller pieces -- or, better yet, originally design it using such smaller pieces.
- Describe strategies that are useful in testing and debugging.
- Write clear comments that communicate to the reader what a function expects, its side-effects if any, and what it returns.

CS Program Learning Outcomes that this course addresses:

This course contributes to departmental learning outcomes of:

- Computational Thinking
- Technical Writing
- Communicating and Collaborating

This course addresses computational thinking at an introductory level, introducing fundamental computing concepts. It addresses technical writing at an introductory level via program documentation and coding standards that stress reusable code, and it addresses communicating and collaborating at an introductory level via experience pair-programming in course lab sessions.

Cal Poly Humboldt Learning Outcomes that this course addresses:

This course contributes to Cal Poly Humboldt learning outcomes (https://academicprograms.humboldt.edu/content/undergraduate-institutional-learning-outcomes) of:

* Some of these are adapted from the ACM Computer Science Curriculum 2001, available from link at: https://www.acm.org/education/curricula-recommendations

- Critical Thinking
- Written Communication
- Quantitative Reasoning

Required Course Materials

- "How to Design Programs", Second Edition, Felleisen, Findler, Flatt, and Krishnamurthi, available on-line, at: https://htdp.org/2024-11-6/Book/index.html
  - (you can also click on the book's title in the book cover image at https://htdp.org/ and follow "The Book" link)
- Turning/EchoEngage Account License used with PointSolutions app (see "Clicker Questions" section below)
  - The PointSolutions app is free, but you do need to purchase a Turning/EchoEngage Account License and register it from the CS 111 course Canvas site, or I will not be able to "see" your answers.
  - Note: I am told that the best price for the Turning/EchoEngage Account License is available when you follow the "Turning Account Registration" link in Canvas -- this link is on the left-hand-side of the course Canvas site.
- Any additional required readings will be made available either on-line, or via resources available through the Cal Poly Humboldt Library such as the ACM Digital Library and Safari TechBooks Online.

Optional Reference Texts

- Runestone's "How to Think Like a Computer Scientist - C++ Edition", https://runestone.academy/runestone/books/published/thinkcpp/index.html
  - **NOTE**: This includes examples that **do not meet class style standards**, but it is still a good reference, and the interactive examples make this especially useful.
- "Problem Solving with C++", Savitch, Addison-Wesley
  - Included as a "recommended text" (in Cal Poly Humboldt bookstore terms) JUST for C++ language reference, for students desiring such a reference. **No** assignments or required readings will come from this text.
  - The current edition is the 10th edition, but for our purposes, the 6th, 7th, 8th, and 9th editions are also fine.

Course Software

We will be using two programming languages during this course – Racket and C++.

Environments for these will be available on https://vlab.humboldt.edu.

Users who install software on their own laptops or desktops are responsible for maintaining their own installations – neither I nor Cal Poly Humboldt computer support can provide tech support for your personal computers, although I will try to answer questions if I can.

Racket

We will be programming in teaching subsets of Racket within the DrRacket programming environment. DrRacket is installed on https://vlab.humboldt.edu.

You may also freely download and install the DrRacket software for Windows, Apple, and Linux by visiting the website https://racket-lang.org and clicking the **DOWNLOAD** button.

C++

We will be using C++ using a CS50 adaptation of Visual Studio Code Codespaces. This runs in a browser, and is available by following the "Log in via GitHub" link from https://cs50.dev. We will walk through how to set up and use this in a CS 111 lab session.

Grading Breakdown

If you are a Computer Science (CS) or Software Engineering (SE) major, note that you must earn at least a **C-** in CS 111 for this course to count towards your major.

Your semester grade will be determined by the percentage of points that you earn in four grading categories, **subject to some minimum requirements**. Here are those grading categories and their grade percentages, followed by those minimum requirements:

Homework assignments:		30%	Note: **NO** homework grades are dropped
Lab exercises:		15%	Note: Lowest **two** lab exercise grades are dropped
Clicker questions:		15%	**Sum** of points earned from answering clicker questions, up to a **maximum** of 120 points
Exams:	**Exam 1:**	10%	**Tuesday, October 14**
	Exam 2:	10%	**Thursday, November 13**
	Final Exam:	20%	**Thursday, December 18, 8:00 am - 9:50 am in NR 101**

In addition, the instructor may, *at their discretion*, issue a **non-passing semester grade** for the course if:

- the semester average for **Homework assignments** falls **below 60%**, or
- the semester average for the three course **Exams** falls **below 60%**

That is, if either of the above minimum requirements is not met, then a semester grade of D or F may be assigned, even if the overall grade average is above 70%.

The purpose of these minimum requirements is to prohibit the practice of simply ignoring part of the course requirements with the thinking that the other parts will be enough to pass.

- In particular, exams are intended to give you an opportunity to demonstrate that you understand at least a minimal reasonable level of the most important course concepts.
- And, because there are hands-on skills that are part of this course that are not tested as effectively on exams, homeworks are intended to give you an opportunity to demonstrate at least a minimal level of programming experience in addition to course concept mastery.
  - (Also, part experience has shown that, in general, students who do not put a solid effort into course homework assignments do not do well on course exams.)

Participation in all aspects of the course must be maintained at acceptable levels in order to learn this material!

So, your semester grade will be computed as shown in this table:

Overall Percentage (based on the given weights)	**Exams Average**	**Homework Average**	**Letter Grade**
>= 93	>= 60	>= 60	A
>= 90 and < 93	>= 60	>= 60	A-
>= 87 and < 90	>= 60	>= 60	B+
>= 83 and < 87	>= 60	>= 60	B
>= 80 and < 83	>= 60	>= 60	B-

Overall Percentage (based on the given weights)	**Exams Average**	**Homework Average**	**Letter Grade**
>= 77 and < 80	>= 60	>= 60	C+
>= 73 and < 77	>= 60	>= 60	C
>= 70 and < 73	>= 60	>= 60	C-
>= 70	< 60	any	D+
>= 70	any	< 60	D+
>= 67 and < 70	any	any	D+
>= 60 and < 67	any	any	D
< 60	any	any	F

More Coursework-related Policies

- It is nearly impossible to write unambiguous specifications. If you have questions about what is being asked for -- whether on a homework problem, in a lab exercise, on an exam question, or even for a clicker question -- you are expected to **ask** me.
  - Being able to ask such questions is a necessary and important real-world skill in computer science!
- There is more to a computer command, expression, statement, function, file, or program than simply whether it "runs".
  - Part of your grade may be determined by how well your work **meets the stated requirements**.

    Your work is expected to meet stated requirements precisely. When working as part of a team on larger software projects, following specifications precisely is vital, and can mean the difference between a working product and one that just sits there.

    Work that is too far from meeting the stated requirements may be returned to you ungraded/not accepted for credit.
  - Work may be graded on **style** as well -- following style and coding standards likewise helps to result in programs that are more readable, understandable, and maintainable over time. Discussions on style will be ongoing throughout the semester.
  - Because you will be learning **good problem-solving practices** in this course as well as programming syntax, you may also be graded on whether you are following these practices (following the **design recipe**, including required documentation and tests, etc.). A program that runs but omits these parts may lose **substantial** credit.

Homework Assignments

- Note that **no homework assignment grades are dropped**; *every* homework assignment grade is included in determining the homework portion of your semester grade. Every homework includes important practice of course fundamentals.
- Homework problems are to be completed individually (although *discussing* homework problems with other students without copying their comments or code is fine!).
- Each homework assignment must be submitted as specified on its handout to be accepted for credit. This may vary for different homework assignments.

- Each homework assignment will be clearly marked with one or more due dates/deadlines (a single homework assignment could have multiple parts with multiple due dates/deadlines).
  - To best benefit from this class, it is important to **practice programming regularly** and to **attempt homework problems before the homework deadlines**.
    - If I notice that a class member is not submitting attempts at homework problems on a timely and regular basis, I may e-mail that class member and require that they set up a meeting with me to discuss this.
  - **If you have attempted all of a homework's short-answer question problems *AND* submitted initial attempts at some of a homework's *programming* problems by its deadline,** you can still submit versions for other of its programming problems, improved versions of its programming problems, (and requested revisions, if any) up until example solutions are posted, before each Exam.
  - Once a homework's example solutions are posted, no more submissions or revisions will be accepted for that homework's programming problems (*unless* you have discussed your unusual situation with me and we have set up a different arrangement).
- You may submit **multiple versions** of homework files and problems; I will grade the **most recent able-to-be-accepted** submission unless you inform me otherwise. (Homework short-answer questions answered on Canvas are handled differently, though -- see the section below.)
  - One reason for encouraging multiple submissions is to encourage you to **turn work in early and regularly**, even perhaps while it is still in-progress, since you can always turn in an improved version later, or if further inspiration strikes, etc.
  - Another benefit of early and regular submissions as you work through homework problems: you don't have to worry about forgetting to submit something that has already been submitted!

Homework Short-Answer Questions

- Most homework assignments will start with one or more problems that are short-answer questions answered on Canvas -- these are meant to give you a chance to see if you are on the right track on new syntax, new terms, or just important concepts.
- These short-answer questions are automatically graded -- after you have attempted all of a problem's questions, you will not be shown the correct answers, but you will be shown if your answers were correct or not, often including some additional explanation.
- **You can attempt these short-answer questions as many times as you would like -- your score for these will be the highest score from all of your attempts.**
  - Because students in the past mentioned that these were useful for exam review, these will be left available/open through the Final Exam.
  - HOWEVER, you will receive the **MOST** benefit from these if you start attempting them well **before** that homework's deadline, as a warm-up to the programming problems.
  - If I notice that a class member is not attempting these homework short-answer questions on a timely and regular basis, I may e-mail that class member and require that they set up a meeting with me to discuss this.

Lab Exercises

- Review and/or lab-exercise-related clicker questions and graded lab exercises will be given during most lab sessions.
  - You **must be present *during* your lab section** -- either in BSS 313 or in the lab Zoom session, depending on your lab section -- to be able to receive credit for that week's lab exercise (unless noted otherwise on a particular week's lab exercise).
  - When a lab exercise specifies that pair-programming is to be used, you **must participate appropriately *in a pair* during your lab section**, either in BSS 313 or in a Zoom breakout room, depending on your lab section, to

be able to receive credit for that week's lab exercise.

- If you miss a lab session, typically its clicker questions and graded lab exercise cannot be made up later (except for extenuating circumstances - please let me know!). However, the **two lowest lab exercise grades** will be dropped from the lab exercise portion of your semester grade.
- You will typically be **pair programming** for lab exercises -- in pair programming, two programmers work on and view the same file at the same time, one typing and the other saying what to type, both looking at the **same** screen, and both also discussing along the way.
  - **Both** are **actively** involved in the programming process **together**.
  - This software engineering practice can result in programs with fewer errors, amongst other potential benefits.
  - While learning to program, this practice can also give you more chances to discuss course concepts with other students, (along with the practical benefit of reducing the total number of questions the instructor has to try to answer during lab sessions, hopefully also reducing your wait time for those answers).
- Note: if, for example, there is an odd number of students at a particular lab, or there are technical difficulties, we'll also sometimes have trios -- in that case, **all three** are still working on and viewing the same file at the same time, one programmer types, and the other two alternate saying what to type, all three are looking at the **same** screen, and of course all are still also discussing along the way.
- It is **not acceptable** to simply sit back during a lab exercise and have your partner do all the typing and saying what to type and discussing -- you are expected to **actively participate** in your pair.
  - Likewise, it is **not acceptable** to work on the lab exercise *individually* before starting to work in a pair.
- Please let me know of any issues that come up related to pair programming, so we can work together to come up with means for dealing with them.
- Once you have completed a lab session's lab exercise, made sure that both of you have a copy of its files, and submitted your copy of those files, it is acceptable to leave the lab session.
  - After completing and submitting the lab exercise, it is also fine to use the remaining lab time to work on the current course homework assignment, to practice course concepts, and/or to ask questions about course-related topics.
  - However, note that questions from those still working on the lab exercise will be prioritized!

Clicker Questions

We will be using the Echo360 (formerly Turning Technologies) PointSolutions student response software in class. There is significant literature indicating that using such so-called "clicker questions" may increase student engagement and success in learning.

Students purchase a Turning/EchoEngage Account license/subscription and register it from the CS 111 course Canvas site, and they use this license with the PointSolutions application on a mobile device or from a web browser. You then will answer questions using this during **every** class meeting (lectures **AND** labs). (Part of the idea here is to stress that **every** class meeting is important, and that participating during **every** class meeting is important.)

Follow the **"Turning Account Registration"** link on the course Canvas site for registering so that your answers receive credit. (You can also purchase the Turning/EchoEngage Account license via this link, and I am told they offer the best price for this.)

This software will be used for in-class questions, which might be asked at any time within class meetings. These will usually be given in a **think-pair-share** fashion, in which you answer a question first on your own, and then discuss your answer with other students, discussing **why** you think your answer is correct; if they gave a different answer, you try to persuade them that yours is the correct answer, and then either of you can change your answer if you wish. The response system will record the overall class response percentages as well as keep track of individual answers.

Note that a large part of the benefit of this is from these discussions with other class members -- research suggests both that putting concepts into your own words helps you to learn them better and that the other class member's explanations may also help you to learn them better.

Typically, you will receive:

- **1.5 points** for a correct answer,
- **0.75 points** for an incorrect answer, and
- **0 points** for no answer,
- but with a **maximum-possible** semester clicker-questions grade of **120**.
- (There may be some no-point questions from time-to-time as well -- these will be noted if/when they come up.)

Thus you will be rewarded for regular attendance and participation.

I hope to run tests of the system during the first week's class meetings, and to begin asking questions that "count" during the second week's lecture meetings. So, you need to purchase and register your license as soon as possible. If there is an issue with this, please let me know as soon as possible.

Finally, **NOTE** that use of another CS 111 student's account, or having someone else use your PointSolutions/Turning/EchoEngage account in a CS 111 class session, or otherwise having anyone but yourself answering a clicker question on your behalf -- that is, pretending that someone is in class who actually is not -- is considered to be **cheating**, with the same policies applying as would be the case if you turned in someone else's work as your own or permitted someone else to copy your work. Please **ASK ME** if you are not sure what I mean by this.

Can clicker questions be "made up" outside of class sessions/lab sessions?

The general rule is that, if you miss a class session, you miss that day's clicker questions, and in general cannot make them up. (But I am willing to discuss alternate arrangements for extenuating circumstances -- contact me sooner rather than later if you would like to discuss such possibilities!)

There will be a sufficient number of questions asked during the semester (at least 120 points worth of questions) to allow for both the possibility of extra credit (up to a **maximum** clicker grade of **120**) or to make up for a day that you are out for illness (although note that you are still responsible for finding out what you missed on such days).

Exams

There will be two exams during the semester and a Final Exam, at the dates given below.

Make-up exams are only possible by special prior arrangement or because of extenuating circumstances. You are expected to **contact me as soon as reasonably possible** in such circumstances.

There will be a review session before each of these exams as noted in the Tentative Course Schedule section.

Exam 1:

Exam 1 will be given during class on **Tuesday, October 14 in NR 101**.

Exam 2:

Exam 2 will be given during class on **Thursday, November 13 in NR 101**.

Final Exam:

The Final Exam will be given **in NR 101** from **8:00 am - 9:50 am** on **Thursday, December 18**. (This is the required date and time specified in the campus Final Exam schedule for a course that meets at 9:00 am on Tuesdays and Thursdays.)

NOTE: You can also find the schedule for ALL of YOUR Final Exams in your Student Center! See:
https://studentcenterhelp.humboldt.edu/final-exam-schedule

IMPORTANT COVID-Related Information

NOTE: This information is subject to change at any time as the university responds to the changing profile of COVID. It is the student's responsibility to be current on all university regulations regarding COVID as they are changed and updated.

Students are required to comply with all university regulations regarding COVID.

Testing Positive

If you test positive for COVID, you should not come to class. You should notify me (and your other face-to-face course instructors), and follow campus Health Center and/or your medical provider's advice regarding when to return to class.

Please notify me again when you have recovered so that we can make a plan for you to get caught up with the class.

Thanks for your cooperation to keep everyone safe and our course on track this semester!

Other Expectations of the Student

- Read this syllabus, and be prepared to verify in a required Canvas activity that you have received it, have read it, and understand its contents.
- **Attend all class sessions, and participate!** Participating includes:
  - paying attention
  - discussing clicker question answers and class concepts with other students
  - being an attentive partner when pair-programming in lab
  - asking questions
- **Follow the design recipe steps, in order,** for *every* function you write.
- There is a general rule-of-thumb for college-level courses:

  To be successful in a course, you should plan to spend at least 2 hours outside of class for each 1 hour of college course credit. That implies an estimate of **at least 8 hours a week spent outside of class for this 4-credit course.**
  - You can only learn programming by **practicing** it. Practicing programming as much as possible helps!
  - This can include **typing in and playing around with in-class examples**, experimenting to see if something you are curious about really works like you think, and so on.
  - Think of a musical instrument -- you have to practice to master playing a guitar, violin, trumpet, drums, etc. You can't master it by just reading about the instrument. Think, also, of sports skills such as pitching, archery, etc. -- again, repetition and practice is required to hone such skills.
- Complete reading assignments in a timely fashion. Ask me if you have any questions about them.
- Check the CS 111 public course website and Canvas course site **frequently** for homework and other assignments, postings of course handouts and in-class examples, announcements, and updates.
- Check your Cal Poly Humboldt e-mail **daily** Monday through Friday.
  - All e-mails that I send for this course will include **`CS 111`** in their `Subject:` line.
  - Likewise, include **`CS 111`** along with a **description** of your e-mail in the `Subject:` line of all class-related e-mails that you send to me.
- Start working on homework assignments as soon as they are posted, submitting frequently. This gives you time to ask questions if you run into problems.
  - Why spend 4 hours struggling with a problem the night before the homework assignment is due, when you can spend 10 minutes composing an e-mail early in the week, work on other problems while waiting for the answer,

and then get a reply that makes everything clearer as soon as you read it?

- Ask questions when you are having difficulty understanding a class concept or not making progress on a homework problem.
  - Ask questions early and often (I will gently let you know if you are overdoing it.)
  - Writing programs can be a notorious time-eater. Sometimes a very small issue can take a long time to locate and fix, especially if you do not ask for help.
  - Later concepts are built upon earlier concepts as the course progresses -- if you ask as soon as you realize that some concept is not clear to you, that can help keep you from falling behind.
- Keep backups of your CS 111 files; if I cannot open one of your submitted lab exercise or homework files, I may need you to re-submit it or to e-mail it to me.
- If you have not completed a lab exercise or homework problem by its deadline, submit whatever you have done up to that point, even if it is not complete.
  - Remember, as noted earlier in this syllabus, submitting *something* by a homework's deadline gives you the possibility of submitting improved versions and attempts at other of its problems after the deadline, up until example solutions are posted, before each Exam.
  - Submitting what you have by the deadline shows that you have started, and *might* allow me to give you feedback based on what you have done so far.
  - Note that significant credit is given for following the design recipe steps, to encourage good programming habits.
  - I believe in partial credit on homeworks, believing that if you have at least started working on a problem, any eventual posted example solution will be more helpful/understandable than if you have not.
- Take the opportunity to learn how to write your own thoughts; don't plagiarize. Be sure to give credit where credit is due and cite your sources.
- If example solutions for selected homework problems are posted, read those over and compare them to how you approached those problems. Be sure to ask me if you have any questions as a result!
- When grades are posted to the course Canvas website, check them and let me know of any discrepancies or issues.

Class Culture*

Course expectations may evolve during our experiences together in class during the semester, but some of the important guiding principles include:

- Respect for each other (what does that mean to you?)
- Come to class sober
- Keep cell phones and other distractions put away
- Be in the classroom before class starts, so that you're ready when it starts
- If you need to leave in the middle of class, do so as quietly and unobtrusively as you can
- Stay until class is over
- Be a regular and willing participant

*Contact the instructor if you need special accommodation or exception from these rules.

Expectations of the Instructor

- I will prepare and review course materials to be as current and accurate as possible.

- I will be available to answer questions or issues that may arise for you during this course. **Expect a 24-hour turnaround time for response to e-mails on weekdays and 48 hours on weekends.**
- I will try to the best of my ability to prepare you for the assignments and other assessments in this course.
- I will utilize fair and honest evaluation techniques for each assignment required for this course.
- I will do my best to address the needs of a diverse range of learning styles in this course.
- I will only share your student information per FERPA (federal privacy) guidelines.

Other Course Policies

Inclusivity

Students in this class are encouraged to speak up and participate in-class. Each of us must show respect for each other because our class represents a diversity of beliefs, backgrounds, and experiences. I believe that this is what will enrich all of our experiences together. I recognize that our individual differences can deepen our understanding of one another and the world around us, rather than divide us. In this class, people of all ethnicities, genders and gender identities, religions, ages, sexual orientations, disabilities, socioeconomic backgrounds, regions, and nationalities are strongly encouraged to share their rich array of perspectives and experiences.

If you feel your differences may in some way isolate you from our classroom community or if you have a specific need, please speak with me early in the semester so that we can work together to help you become an active and engaged member of our class and community. *(Adapted from Cal Poly Humboldt Canvas Accessible Syllabus Template, which was in turn adapted from CSU Chico and Winona State University)*

Thus, spoken language and body language should emanate respect for everyone in our classroom community. This includes coming to class on time and being prepared to listen and share. (*Adapted from Jayne McGuire's syllabi language*)

CS 111 E-mail Policies

- **NOTE: do NOT use Canvas messages** to contact me or ask me a question -- send me **actual e-mail messages** instead. Handling Canvas messages is time-consuming and error-prone on my end.
  - Please ASK me if you are not sure what I mean by this.
- Students are responsible for checking their Cal Poly Humboldt e-mail account for official communications. You are expected to check for course-related messages as well.
  - While students may elect to redirect messages sent to their official Cal Poly Humboldt e-mail address to another address, those who redirect their e-mail to another address do so at their own risk.
  - Cal Poly Humboldt E-mail Policy: https://policy.humboldt.edu/p21-01-email-policy
- All e-mails that I send for this course will include **`CS 111`** in their `Subject:` line.
- Likewise, include **`CS 111`** along with a description of your e-mail in the `Subject:` line of all class-related e-mails that you send to me.
  - This will help your e-mail be more recognizable as a class-related message, and will make it less likely that I will accidentally overlook it.
- Remember to include a **descriptive subject** along with the `CS 111` in that `Subject:` line -- this also increases the chances that I will notice and reply to your question more promptly.
  - (In particular, do not just reply to a class e-mail message I have sent previously, and do not simply leave the `Subject:` line blank!)
- Ask **specific** questions via e-mail -- for less-specific or broader questions, come to student hours or make an appointment to meet with me. Overly-vague or broad questions are problematic to answer by e-mail.

- For example, an example of a **specific** question is:

  "When I try to run my function `myfun` [attach your file with `myfun`'s signature, purpose statement, tests, and code], I receive the following error message: [paste in the first 4-5 lines of that error message]

  Can you point me in the right direction about what is wrong?"
- An example of an overly-vague or broad question is:

  "Here's my program: [just pasting in its code]. Is it right?"

- When e-mailing a question about a program,
  - attach a copy of your program file(s)

    and ALSO
  - paste in the first 4-5 lines of the **error messages** you are getting

    and/or descriptions of **bizarre behavior** you are seeing.
- It is perfectly reasonable if you e-mail me a specific question and then happen to find out the answer yourself before you receive my answer. (Letting me know you've found the answer is fine, too!)
- Likewise, it is not a problem if you happen to send me several specific questions in separate e-mails (for example, as you work on different homework problems while awaiting earlier answers). I can answer shorter e-mails more quickly than longer e-mails.
- Expect a **24-hour turnaround time** for response to e-mails on weekdays and **48 hours** on weekends.
  - So, in general, if I have not replied to your e-mail within 24 hours, please **re-send** it, just in case I have overlooked it or some glitch occurred.
  - (And if there seems to be a chance that your message is getting chomped by a spam filter -- rare, but not unprecedented! -- leave me a message at 707-826-3381 with the `Subject:` line of the e-mail you are trying to send and the e-mail address you are using, and I will see if I have indeed received it!)
- You are expected to **sign** each e-mail you send me with **your name** -- sometimes the sender's identity is not obvious from one's e-mail address, especially for an off-campus e-mail address.
- Please take a few minutes to ensure that your message reflects a professional tone. I know I have sent an e-mail or two in the heat of the moment that I soon regretted. Take your time and communicate professionally. (*Adapted from Jayne McGuire's syllabi language*)

Course Absences

Between the ample quantity of clicker questions asked during the semester, and the two dropped lab exercise grades, you can be absent several times from non-exam lecture or lab sessions without significant direct penalty, for whatever reason. However, it is **your responsibility** to find out what was announced and covered on those days; "I wasn't there that time" is not an acceptable excuse.

Please let me know if class or life issues are making it difficult for you to attend class meetings or to keep up with course material and coursework, so we can make arrangements to help you work through those. It helps if you let me know **sooner** rather than later about such issues!

Academic Honesty, Plagiarism, and Generative AI Tools

Students are responsible for knowing Humboldt's Student Academic Honesty Procedure policy, available at:

https://www2.humboldt.edu/studentrights/academic-honesty

Plagiarism is a serious offense. Copying of another person's work and submitting it as your own for individual assignments, or providing your work to others for them to copy and submit as their own for such assignments, is not acceptable.

Among the actions that are **unacceptable** are submitting another's program, code, or file as your own; giving programs, code, or files to another; and failing to quote material (that includes algorithms, project, code, and comments, too!) taken from another source. This even applies to comments as well as actual executing code.

Likewise, it is **NOT** okay to copy or post homework answers or code from or to an online discussion or from or to sites such as Chegg.

Generative artificial intelligence (AI) programs, such as ChatGPT, may **not** be used to generate any of the work or assignments that you submit as your work in this course. The use of generative AI programs defeats the writing requirements and critical thinking skills that are vital to achieving CS 111's learning outcomes. Submission of partial or complete work from generative AI programs is not permitted and will be treated as plagiarism as defined in Humboldt's Student Academic Honesty Procedure, and handled in accordance with that Procedure. (*Adapted from Humboldt's Center for Teaching & Learning's "Artificial Intelligence Sample Syllabus Statements AY23-24"*)

Learning takes hard work; when students turn in others' work as their own, or provide it to others to copy, it is a slap in the face to those seriously interested in learning who are putting in that effort. Not turning in an assignment results in no credit for that assignment, but that is an honest grade.

Evidence of copying or plagiarism will result in appropriate penalties, up to and including a failing grade in the course.

All of the above said, also note the following:

- In the case of pair programming during weekly lab sessions, you are *intended* to work together, with one student typing the code while all team members collaborate, to create files that then include all of your names. This should mean that you all *participated* in pair-programming for that assignment. This is the ONLY authorized exception to the above policy! (*Adapted from David Tuttle's CS 112 Syllabus*)
- Did you find an interesting inspiration from a Google search or from a book for your algorithm or for a part of your code? **Attribute** it -- include a **comment giving its source**! Then you are *not* "failing to quote material ... taken from another source". (And, of course, *adapt* it to meet class coding and style standards as well as the needs of the particular program.)
- Note that it is fine and *encouraged* to make use of functions and example code from posted in-class examples -- reusing tested and debugged functions is good programming practice! -- but it is good professional practice to comment their source as well. (For CS 111 purposes, this can be as simple as "`from posted CS 111 examples`", for example.)

Note that it is **your** responsibility to ensure that your homework files are read-protected. If you are careless about this, and someone else copies your work, you will share the penalty. (In particular, be very careful about leaving work on shared network drives, or in UNIX/Linux directories that are not read-protected.)

Is is OK to help each other?

On exams, **no**. (That said, *studying* together for each exam, before taking it, is an excellent idea and encouraged!)

For homework assignments, discussing approaches to homework problems is fine -- a good rule-of-thumb is that you are discussing approaches but not writing down or copying how to complete a particular problem.

Students may also help one another in determining causes of homework problem bugs, or in determining the meaning of error messages.

However -- again -- any copying or modifying of someone else's answers, source code, or files, OR of providing answers, source code, or files to another, related to homework assignments and exams is definitely over the line, and never justified.

More on Asking Questions/Getting Help

- You are encouraged to ask questions in class, in student hours, and by e-mail. The most successful students are those who are not afraid to ask questions early and often (I will gently let you know if you are overdoing it).
- Especially with regard to homework assignments, it is usually better to ask a question **sooner** than later.

- For example, it is better to send an e-mail with a specific question you have about a problem as soon as you think of it, rather than wait a day or two until the next class meeting or student hour.
- If you wait to ask such questions, you might not have time to complete the assignment.

Incompletes

Incompletes are rarely given and only in the case of a true emergency. They are not appropriate for students who find they have fallen behind on assignments, missed a test, or taken on too much academic, work, or family responsibilities. For these situations, dropping the course would be appropriate (**if** that is still possible according to the University policies for dropping courses).

If you are facing extenuating or emergency circumstances at any time during the semester, please consider contacting the Cal Poly Humboldt **Campus Assistance, Response, and Engagement (CARE) Services** office:

https://deanofstudents.humboldt.edu/CARE

Campus policies

The following leads to useful links regarding Cal Poly Humboldt policies, procedures, and resources:

https://academicprograms.humboldt.edu/content/syllabus-addendum

All of the policies linked from the above are applicable to this class, and you are expected to be familiar with these policies.

The following are just a FEW highlights from this site, along with a few additional campus-policy-related notes:

Campus Disabilities Resource Center (CDRC)

Persons who wish to request disability-related accommodations should contact the **Campus Disability Resource Center** in Library 005, by phone at **707-826-4678**, or by email at **student504@humboldt.edu**. Disability accommodations must be pre-approved by the Campus Disability Resource Center.

Note that classrooms should be equipped with at least one desk and chair for students with disabilities or temporary physical challenges. To ensure that students who need this equipment are able to use it, if you are not physically challenged yourself, select another seat in the classroom.

You can reach the Campus Disability Resource Center's web site at:

https://www.humboldt.edu/cdrc

Please note that some accommodations may take up to several weeks to arrange. If you are eligible for such accommodations, please contact me as soon as possible to discuss them.

Dropping or Adding a Class

- Students are responsible for knowing the University policy, procedures, and schedule for dropping or adding classes.
  - You can find these deadlines for Fall 2025 by going to:

    https://www.humboldt.edu/featured-events

    ...and then scrolling down to the "**MORE CALENDARS**" section,

    finding the "**ACTIVITIES & DEADLINES**" subsection, and

    clicking on the "**Fall 2025**" link
  - (which leads to: https://www.humboldt.edu/sites/default/files/registrar/2025-08/d-fall-2025.pdf)
    - (There are MANY important deadlines in this calendar -- it is well-worth reading through!)
- **Note that the Add/Drop deadline for Fall 2025 is 11:59 pm on MONDAY, SEPTEMBER 8th.**

  - This is the deadline to add or drop courses through the Student Center.
  - After September 8th, dropping a course requires a "documented serious and compelling reason", and it is the **Registrar's Office** that determines what constitutes a "documented serious and compelling reason".
  - Note that it is the student's responsibility to properly drop a course.
- You can also find more information about dropping or adding a class at:

  https://registrar.humboldt.edu/forms - and click on **Add/Drop Date** on the right-hand side (OR toward the BOTTOM if viewing this on a phone or within a narrow browser window!)
- You can find the University policies for repeating classes at:

  https://registrar.humboldt.edu/forms#policies - and click on **Repeating Courses** on the right-hand side (OR toward the BOTTOM if viewing this on a phone or within a narrow browser window!)

Note about Course Grade Modes

Note that courses applying to CS major/minor or SE major requirements must be taken with a grade mode of **letter grade** (that is, NOT with a grade mode of CR/NC, credit/no credit).

If you are taking this course as a **free** elective, however (and **not** applying it to a CS or SE major or a CS minor), then note that there is a limit of **at most one optional CR/NC course per term**, and that, for Fall 2025, the deadline to change grade modes is **Monday, November 10th.**

For more information on optional CR/NC grade mode, see:

https://registrar.humboldt.edu/node/407 - and click on **Credit Limitations** on the right-hand side (OR toward the BOTTOM if viewing this on a phone or within a narrow browser window!), and within that scroll down to the **Credit/No Credit** section within.

Attendance and disruptive behavior:

Students are responsible for knowing policy regarding attendance and disruptive behavior:

https://www2.humboldt.edu/studentrights/attendance-behavior

- **Class disruption:** University policy requires that instructors eliminate disruptions to the educational process. Distractions such as excess talking or behaviors that disrupt the class are not acceptable.
  - Students indulging in such behaviors will first be warned before any additional measures are taken (although a warning is not required in the case of abusive behavior).

In Case of Emergency:

Adapted from "**Earthquake Resources for HSU Faculty and Staff**", by the Cal Poly Humboldt Geology Department:

IN THE EVENT OF AN EARTHQUAKE:

- DROP AND COVER YOUR HEAD, AS BEST YOU CAN.
- DO NOT RUN, JUMP OVER SEATS, PUSH ANYONE, OR HEAD FOR THE DOORWAY.
- STAY WHERE YOU ARE UNTIL THE SHAKING STOPS.
- WE WILL CALMLY AND QUICKLY EXIT THE CLASSROOM/LAB AND ASSEMBLE AT OUR ASSIGNED EMERGENCY ASSEMBLY POINT (EAP).
- Download the MyShake app on your smartphone so that you can receive earthquake alerts from the earthquake early warning system.

Want to read more from this excellent resource? Here is a link (provided with permission from Professor Melanie Michalak):

https://docs.google.com/document/d/1_TeV_SoN2M7jqUyOVQPdSsw5jndGMY1zuCJroFUF8zM

More generally:

Emergency Information

Please review the evacuation plan for the classroom (posted on the orange signs). During an emergency, information regarding campus conditions can be found at **707-826-INFO** or:

https://www.humboldt.edu/emergency

TENTATIVE Course Schedule: (subject to change with fair notice)

- Note also that **additional readings may be added** to those given below.

Week 1: August 26, 28, 29

- **Topics**: Introduction to course; practice clicker questions; simple and compound expressions; data types (number, boolean, string, image); syntax and semantics
- **Reading**:

  from public course web site and Canvas site:
  - Course **Syllabus**

  from "How to Design Programs", 2nd Edition (HtDP/2e):
  - **Preface** - JUST the 1st part, up until **"Systematic Program Design"**

    direct link: https://htdp.org/2024-11-6/Book/part_preface.html
  - **Prologue: How to Program** - JUST the first part, through the "Arithmetic and Arithmetic" section (up to but not including **"Inputs and Output"**)

    direct link: https://htdp.org/2024-11-6/Book/part_prologue.html
  - Section I - Fixed Size Data - **Chapter 1 - Arithmetic** - at least the sections **1.1 - 1.5 and 1.7**

    direct link: https://htdp.org/2024-11-6/Book/part_one.html#%28part._ch~3abasic-arithmetic%29
- **Homework 1 out**

Week 2: September 2, 4, 5

- **Topics**: Introduction to designing your own functions and to the program design recipe; parameter variables
- **Reading**: from "How to Design Programs", 2nd Edition (HtDP/2e):
  - **Prologue: How to Program** - the section **"Inputs and Output"**

    direct link: https://htdp.org/2024-11-6/Book/part_prologue.html#%28part._some-i%2Fo%29
  - Section I - Fixed Size Data - **Chapter 2 - Functions and Programs** - at least the sections **2.1**, **2.2**, and **2.4**

    direct link: https://htdp.org/2024-11-6/Book/part_one.html#%28part._ch~3afuncs-progs%29
  - **Preface** - the section "**Systematic Program Design**"

    direct link: https://htdp.org/2024-11-6/Book/part_preface.html#%28part._sec~3asystematic-design%29
  - Section I - Fixed Size Data - **Chapter 3 - How to Design Programs** - at least the sections **3.1** and **3.3 - 3.5**

    direct link: https://htdp.org/2024-11-6/Book/part_one.html#%28part._ch~3ahtdp%29
- **Homework 1 due - 11:59 pm Friday, September 5**
- **Homework 2 out**

Week 3: September 9, 11, 12

- **FYI: NOTE: Last day to drop a course through your Student Center (without a W and without a serious and compelling reason) is MONDAY, September 8.**
- **Topics**: Continuing introduction to designing your own functions and to the program design recipe; intro to `2htdp/universe`'s `big-bang` function; composing functions and definitions to create programs
- **Reading**: from "How to Design Programs", 2nd Edition (HtDP/2e):
  - Section I - Fixed Size Data - **Chapter 2 - Functions and Programs** - sections **2.3** and **2.5**

    direct links:

    https://htdp.org/2024-11-6/Book/part_one.html#%28part._sec~3acomposing%29

    https://htdp.org/2024-11-6/Book/part_one.html#%28part._sec~3aprogs%29
  - Section I - Fixed Size Data - **Chapter 3 - How to Design Programs** - sections **3.4** and **3.6**

    direct links:

    https://htdp.org/2024-11-6/Book/part_one.html#%28part._sec~3adesign%29

    https://htdp.org/2024-11-6/Book/part_one.html#%28part._.D.K._sec~3adesign-world%29
- **Homework 2 due - 11:59 pm Friday, September 12**
- **Homework 3 out**

Week 4: September 16, 18, 19

- **Topics**: Boolean functions, conditional expressions, conditional functions, enumerations and intervals, adding the template step to the program design recipe
- **Reading**: from "How to Design Programs", 2nd Edition (HtDP/2e):
  - Section I - Fixed Size Data - **Chapter 4: Intervals, Enumerations, Itemizations** - sections **4.1 - 4.4**

    direct link: https://htdp.org/2024-11-6/Book/part_one.html#%28part._ch~3aintervals-enums%29
- **Homework 3 due - 11:59 pm Friday, September 19**
- **Homework 4 out**

Week 5: September 23, 25, 26

- **Topics**: Itemizations; introduction to lists
- **Reading**: from "How to Design Programs", 2nd Edition (HtDP/2e):
  - Section I - Fixed Size Data - **Chapter 4: Intervals, Enumerations, Itemizations** - sections **4.5 - 4.6**

    direct link: https://htdp.org/2024-11-6/Book/part_one.html#%28part._itemization._sec~3aitemization%29
  - **Section II** - Arbitrarily Large Data - **Chapter 8: Lists** - sections **8.1 - 8.4**

    direct link: https://htdp.org/2024-11-6/Book/part_two.html#%28part._ch~3alists1%29
  - Section II - Arbitrarily Large Data - **Chapter 9: Designing with Self-Referential Data Definitions** - the **beginning section**, up to the beginning of section 9.1

    direct link: https://htdp.org/2024-11-6/Book/part_two.html#%28part._ch~3adesign-lists%29
- **Homework 4 due - 11:59 pm Friday, September 26**
- **Homework 5 out**

Week 6: September 30, October 2, 3

- **Topics**: More on lists; simple examples of file input/output
- **Reading**: from "How to Design Programs", 2nd Edition (HtDP/2e):
  - Section I - Fixed Size Data - **Chapter 2 - Functions and Programs** - sections **2.3** and **2.5**

    direct links:

    https://htdp.org/2024-11-6/Book/part_one.html#%28part._sec~3acomposing%29
    https://htdp.org/2024-11-6/Book/part_one.html#%28part._sec~3aprogs%29
  - **Section II** - Arbitrarily Large Data - **Chapter 10 - More on Lists -** sections 10.1 and 10.3, and Figure 67

    direct links:

    https://htdp.org/2024-11-6/Book/part_two.html#%28part._sec~3alist-produce%29
    https://htdp.org/2024-11-6/Book/part_two.html#%28part._sec~3alis-lis%29
    https://htdp.org/2024-11-6/Book/part_two.html#%28counter._%28figure._fig~3aread%29%29
- **Homework 5 due - 11:59 pm Friday, October 3**
- **Homework 6 out**

Week 7: October 7

- **Tuesday, October 7 - REVIEW for Exam 1** (NOTE that this will include several review clicker questions.)
- **October 10, 11 - no lecture or labs**, because instructor will be traveling to a conference (CCSC-NW 2025)
- **Homework 6 due - 11:59 pm FRIDAY, October 10**

Week 8: October 14, 16, 17

- **Tuesday, October 14 - Exam 1**
- **Topics (after Exam 1)**: Introduction to C++: simple expressions, compound expressions, data types, and functions; definition of a C++ program
- **Reading:** from Canvas module "C++ Transition Readings":
  - **"Some Racket/C++ Comparisons"**
  - **Section 1 - "Numbers, Expressions, Simple Programs"** - Preface and all sections
- **Homework 7 out**

Week 9: October 21, 23, 24

- **Topics**: more on C++ compound expressions, `main` functions, and non-`main` functions, and C++ version of the design recipe
- **Reading:** from Canvas module "C++ Transition Readings":
  - **Section 2 - "Function and Variable Definitions = Programs"** - all sections
  - **Section 4 - "Writing your own C++ programs"** - just section 4.3 - "The C++ `main` function"
- **Homework 7 due - 11:59 pm Friday, October 24**
- **Homework 8 out**

Week 10: October 28, 30, 31

- **Topics**: intro to concepts of C++ class, method; start intro to conditional statements in C++; (if time) examples of recursion in C++
- **Reading:**

  from Wikipedia:

  - **"Method (computer programming)" -** https://en.wikipedia.org/wiki/Method_(computer_programming)

  from Canvas module "C++ Transition Readings":

  - **Section 3 - "Conditional Expressions and Functions"** - all sections
- **Homework 8 due - 11:59 pm Friday, October 31**
- **Homework 9 out**

Week 11: November 4, 6, 7

- **FYI: NOTE: Last day to withdraw from a course with a W, with a serious and compelling reason, and subject to your 18 semester-units drop limit is Monday, November 3.**
- **Topics:** intro to `switch` statement; introduction to `void` functions; introduction to local variables, mutation, and assignment statements; scope; introduction to `cin` and `getline` (interactive input)
- **Thursday, November 6 - REVIEW for Exam 2** (NOTE that this will include several review clicker questions.)
- **Reading:** from Runestone's "How to Think Like a Computer Scientist - C++ Edition",

  https://runestone.academy/runestone/books/published/thinkcpp/index.html:

  - **Section 13.2 - `switch` statement**

    direct link: https://runestone.academy/runestone/books/published/thinkcpp/Chapter13/switch_statement.html

  from Canvas module "C++ Transition Readings":

  - **Section 4 - "Writing your own C++ programs"** - sections 4.1, 4.2, 4.4, and 4.5

  from Runestone's "How to Think Like a Computer Scientist - C++ Edition",

  https://runestone.academy/runestone/books/published/thinkcpp/index.html:

  - **Section 2.1 - More Output** - has some useful `cout` practice exercises

    direct link: https://runestone.academy/runestone/books/published/thinkcpp/Chapter2/MoreOutput.html
  - **Section 2.3 - Variables**

    direct link: https://runestone.academy/runestone/books/published/thinkcpp/Chapter2/Variables.html
  - **Section 2.4 - Assignment**

    direct link: https://runestone.academy/runestone/books/published/thinkcpp/Chapter2/Assignment.html
- **Homework 9 due - 11:59 pm FRIDAY, November 7**

Week 12: November 13, 14

- **FYI: NOTE: Last day to change a registered class' grade option to CREDIT/NO CREDIT is Monday, November 10.**
  - (limit of at most **one** optional CR/NC course permitted per term)
  - (that said, also note that courses applying to your CS or SE degree requirements must **NOT** be taken as credit/no credit; they must be graded with a **letter grade**)

- **Tuesday, November 11 - NO CLASS - Veterans Day Holiday - campus closed**
- **Thursday, November 13 - Exam 2**
- **Friday, November 14 -** there WILL be labs, with clicker questions and a lab exercise
- **Homework 10 out**

Week 13: November 18, 20, 21

- **Topics:** intro to (mutation-based) repetition: count-controlled `while`-loops; intro to C++ arrays; adding pseudocode to the design recipe
- **Reading:**

  from Runestone's "How to Think Like a Computer Scientist - C++ Edition",

  https://runestone.academy/runestone/books/published/thinkcpp/index.html:

  - **Section 6.3 - The `while` statement**

    direct link: https://runestone.academy/runestone/books/published/thinkcpp/Chapter6/the_while_statement.html

  from https://runestone.academy/ns/books/published/cpp4python/CollectionData/Arrays.html

  - **Section 5.2. Arrays**
- **Homework 10 due - 11:59 pm Friday, November 21**
- **Homework 11 out**

FALL BREAK - November 25-29

Week 14: December 2, 4, 5

- **Topics**: Some convenient C++ operators: `+=`, `-=`, `*=`, `/=`, and the increment (`++`) and `decrement` (`--`) operators; intro to the `for`-loop; introduction to stream-based file input/output
- **Reading**:
  - for **`for`** loops:
    - from https://www.tutorialspoint.com/cplusplus/cpp_for_loop.htm - C++ for loop
    - from https://runestone.academy/ns/books/published/cpp4python/Control_Structures/for_loop.html - Section 3.3 - For loops
  - from Runestone's "How to Think Like a Computer Scientist - C++ Edition",

    https://runestone.academy/runestone/books/published/thinkcpp/index.html:

    - **Section 7.10 - Increment and Decrement Operators** - direct link: https://runestone.academy/runestone/books/published/thinkcpp/Chapter7/increment_decrement.html
    - **Section 15.2 - Streams**

      direct link: https://runestone.academy/runestone/books/published/thinkcpp/Chapter15/Streams.html
    - **Section 15.3 - File input**

      direct link: https://runestone.academy/runestone/books/published/thinkcpp/Chapter15/File_input.html
    - **Section 15.4 - File output**

      direct link: https://runestone.academy/runestone/books/published/thinkcpp/Chapter15/File_output.html
- **Homework 11 due - 11:59 pm Friday, December 5**
- **Homework 12 out**

Week 15: December 9, 11, 12

- **Topics**: to be determined
- **Thursday, December 11 - Review for Final Exam** (NOTE that this will include several review clicker questions.)
- **Homework 12 due - 11:59 pm Friday, December 12**

Final Exam:

THURSDAY, DECEMBER 18, 8:00 am - 9:50 am in **NR 101**.