Computer Science I

CSCE 155E - Syllabus

School of Computing
College of Engineering
University of Nebraska Lincoln

> "If you really want to understand something, the best way is to try and explain it to someone else. That forces you to sort it out in your own mind... that's really the essence of programming. By the time you've sorted out a complicated idea into little steps that even a stupid machine can deal with, you've certainly learned something about it yourself."
>
> —Douglas Adams, *Dirk Gently's Holistic Detective Agency*
>
> In my experience, you assert control over a computer–show it who's the boss–by making it do something unique. That means programming it... If you devote a couple of hours to programming a new machine, you'll feel better about it ever afterwards"
>
> —Michael Crichton, Electronic Life
>
> The readings were the most underrated things ever (I've never actually read them but I wish I did) thanks for providing them and leaving the whole course available to us.
>
> —Former student a year after taking this course
>
> Enjoyed the videos. Did not really watch them videos until [later], realized that it was a mistake now.
>
> —Former student a year after taking this course

Course Info

Prerequisites: MATH 103 or equivalent.

Description: Introduction to problem solving with computers. Topics include problem solving methods, software development principles, computer programming, and computing in society.

Credit Hours: 3

Postrequisites: The course after this course, CSCE 156 – Computer Science II requires that you receive a grade of C or better in this course to move on. If you are a Computer Science or Computer Engineering major you will need to receive a C or better in this course to continue in the major.

For all other information, see the course website.

University Policies, Resources & Services

Students are responsible for knowing the university policies and be aware of resources found on this page: https://go.unl.edu/coursepolicies

Continuation of Instruction

If in-person classes are canceled by the University during a lecture day, lecture may still be livestreamed. If in-person classes are canceled by the University during a lab/hack day, in-person attendance will not be required, help *may* be made available via zoom and you will still be expected to complete the work remotely.

Skills Objectives

This course has several learning objectives and "skills objectives." These are the skills that, upon successful completion of this course, you should be able to exhibit.

- You should have a mastery of the fundamentals of programming in a high-level language, including data types and rudimentary data structures, control flow, repetition, selection, input/output, and procedures and functions.
- You should be able to approach a reasonably complex problem, design a top-down solution, and code a program in a high-level programming language that automates solutions.
- You should have a familiarity with problem solving methods, including problem analysis, requirements and specifications, design, decomposition and step-wise refinement, and algorithm development (including recursion).
- You should have a familiarity with software development principles and practices, including data and operation abstraction, encapsulation, modularity, code and artifact reuse, prototyping, iterative development, best practices in coding design, style, and documentation, a good understanding of proper testing and debugging techniques and a familiarity with development tools.
- You should have exposure to algorithms for searching, sorting and other problems, graphical user interfaces, event-driven programming, and database access.
- You should have a foundation for further software development and exploration. You should have a deep enough understanding of at least one high-level programming language that you should be able to learn another programming language with relative ease in a relatively short amount of time.

Schedule

See the course website (canvas).

Grading

Assessment (grading) will be based on weekly readings, labs, "hacks", attendance as well as exams and a comprehensive final with the following point distributions.

Category	Number	Points Each	Total
Starter Points			30
PDC Module/Surveys			70
Readings (zyBooks)	13	15	195
Labs	13	15	195
Hacks	14	25	350
Attendace	14	10	140
Exams	2	75	150
Final	1	150	150
Total			1,280

Starter Points

It is important to start out positively. Put yourself in the mindset that you *will* succeed in this course and commit yourself to putting in a full effort in every aspect of it. To get you started, we're giving you **free starter points**. You have a perfect score in this course already! Keep it up!

PDC Modules/Surveys

This course is experimenting with introducing **Parallel and Distributed Computing** (PDC) concepts early in the computing curriculum. These topics are usually covered in depth in advanced elective courses. However, we have developed a series of "codeless" modules that do not require any coding knowledge and cover PDC concepts in an accessible manner.

As part of this study, you will complete pre-module survey(s) which will be graded based on completion. You will then go through several modules on PDC which includes running and observing some simulations, visualizations, etc. You may complete these modules at your own pace. Once completed, you will then take post-module survey(s) again graded based on completion only. Full details and due dates are posted in canvas.

Readings

This course is organized into modules (roughly 1 module per week) and each module has reading associated with it. Some reading is required and other

readings are suggested/recommended and provide a deeper understanding of the topics. Some of the required reading will be assessed and will count toward your grade.

The assessed reading for this course is delivered through an online textbook provided by zyBooks, a web-based interactive text book that has you do some reading and then assesses your understanding through short quiz questions. You may retry these quizzes until you get them correct. These readings are assessed based on completion and are generally due *before* the lectures on the modules in order to better prepare you for the presented material. If you do not complete the readings for each module, you will not receive any credit.

Not every set of required reading represents an equal amount of work. Some modules have more reading than others. However, all assessment is equally weighted since it is based on completion.

Labs

There will be weekly labs that give you hands-on exercises for topics recently covered in lecture. The purpose of lab is not only to give you further working experience with lecture topics, but also to provide you with additional information and details not necessarily covered in lecture. Each lab will have some programming requirements and a supplemental worksheet.

Depending on logistics, those in the on-campus section may be randomly paired with a partner. One of you will be the *driver* and the other will be the *navigator*. The navigator will be responsible for reading the instructions and guiding the driver. The driver will be in charge of the keyboard and will type the code. Both driver and navigator are responsible for developing and working through solutions together. Neither the navigator nor the driver is "in charge," it is an equal partnership. Beyond your immediate pairing, you are encouraged to help and interact and with other pairs in the lab.

Each lab is assessed based on completion. In general you will need to submit code and (possibly) an electronic writeup of your worksheet through an online grading system. You will have until midnight on the day of the lab (Tuesdays) to submit your solutions. Points will be awarded based on the results of the online grader.

Hacks

There will be weekly Thursday *hack sessions* that will provide an opportunity to start working on exercises in an open, collaborative environment. Each hack session is a simple program or exercise. You may not necessarily complete the entire exercise during the hack session.

Further details are provided in the handouts, but you are highly encouraged to collaborate with any individual and to receive as much help as you desire on the exercises.

You may complete hacks on your own, but you are encouraged to pair up and collaborate with (**at most**) one other student. If you choose to pair up with another student, you must form a pair using canvas (People -> Hack Pairs).

Attendance

To ensure participation your attendance **and** engagement in your weekly lab/hack sessions will be graded. Each session will be worth 10 points. To earn the full 10 points you must:

- Attend **both** the lab and hack session for the entire time. Do not be more than 5 minutes late.
- Put forth a full effort. You must work on the lab/hack for that week or for other material in the course; you may not work on other material or remain idle.
- You may only leave the lab/hack session early if you have already completed both the lab and hack for that module.

Alternative

It is understandable that you may not be able to attend every lab/hack session or that your background or approach to this material means you don't need the structured environment that the lab/hack session provides. As an alternative, *if you can prove yourself capable* of completing labs and hacks on your own you may choose not to attend the lab/hack session.

If you attend as outlined above, you get the full 10 points. If you choose not to attend, you may still earn the 10 points ***provided*** that you earn at least 75% of the remaining points for each module (Reading/Lab/Hack). If you do not attend and you earn less than 75% of the points for the module, you will get a zero for the attendance portion of the module.

Exams

There will be several exams as well as a comprehensive final exam. These will be open-book, open-note, *required computer* exams. The exams consist of live coding exercises for which you will need your own machine as you will be coding and submitting programs online for grading. More details will be announced closer to the exam dates. No collaboration is allowed on exams.

15th Week Policy Notification

A per UNL's 15th Week Policy (also known as "dead week") available here:

https://registrar.unl.edu/academic-standards/policies/fifteenth-week-policy/

we are required to serve written notice that the final assignment as well as the final lab, hack, and assignment will be due during the 15th week.

Scale

Final letter grades will be awarded based on the following standard scale. This scale may be adjusted upwards if the instructor deems it necessary based on the final grades only. No scale may be made for individual assignments or exams.

Letter Grade	Percent
A+	>= 97%
A	>= 93%
A-	>= 90%
B+	>= 87%
B	>= 83%
B-	>= 80%
C+	>= 77%
C	>= 73%
C-	>= 70%
D+	>= 67%
D	>= 63%
D-	>= 60%
F	< 60%

Grading Policy

If you have questions about grading or believe that points were deducted unfairly, you must first address the issue with the individual who graded it to see if it can be resolved. Such questions should be made within a reasonable amount of time after the graded assignment has been returned. No further consideration will be given to any assignment a week after it grades have been posted. It is important to emphasize that the goal of grading is consistency. A grade on any given assignment, even if it is low for the entire class, should not matter that much. Rather, students who do comparable work should receive comparable grades (see the subsection on the scale used for this course).

Late Work Policy

In general, there will be no make-up exams or late work accepted. Exceptions may be made in certain circumstances such as health or emergency, but you must make every effort to get prior permission. Documentation may also be required.

Assignments have a strict due date/time as defined in canvas and the online handin system. All program files must be handed in as specified in individual assignment instructions. Programs that are even a few seconds past the due date/time will be considered late and you will be locked out of handing anything in after that time.

Grader Policy

Failure to adhere to the requirements of an assignment in such a manner that makes it impossible to grade your program means that a disproportionate amount of time would be spent evaluating your assignment. For this reason, we will not grade any assignment that does not compile and run through the online grading system.

Academic Integrity

All homework assignments, programs, and exams must represent your own work unless otherwise stated. No collaboration with fellow students, past or current, is allowed unless otherwise permitted on specific assignments or problems. The Department of Computer Science & Engineering has an Academic Integrity Policy. All students enrolled in any computer science course are bound by this policy. You are expected to read, understand, and follow this policy. Violations will be dealt with on a case by case basis and may result in a failing assignment or a failing grade for the course itself. The most recent version of the Academic Integrity Policy can be found at https://computing.unl.edu/academic-integrity-policy

Artificial Intelligence Tool Usage

Various Artificial Intelligence (AI) applications have been developed in recent years (technically called Large Language Models) that can generate code based on user interactions. The use of these tools (including but not limited to ChatGPT, GitHub's co-pilot, etc.) in *any* capacity for this course is ***strictly prohibited*** and any such use will be considered a violation of academic integrity. Violations may include zeros on assignments or failure of the course.

These tools have legitimate uses, but not for students in this course. You are here to learn the basics of problem solving, programming and computing. These tools simply give you answers without you having to think, learn or understand the solution(s) that they give you. Often these tools will give you answers that violate academic integrity even if you do not explicitly ask them to do so. It would *clearly* be a violation of academic integrity if you asked another *human* to write a program for you and they did so. Asking an AI to do so is no different. You will not build muscle by having someone else lift weights for you. You will not learn anything by having someone else, human or AI, do the learning for you.

That said, these tools *can* provide a meaningful learning experience if they are properly filtered. For that reason, we will be allowing the use of the built-in "Rubber Duck" debugger chat bot in the CS50 IDE. It has been configured so that it won't provide code directly but you can ask it questions to understand your code and/or resolve problems.

Communication & Getting Help

The primary means of communication for this course is an online message forum system designed for college courses. You should have received an invitation to join. If you have not, contact the instructor immediately. On this forum, you can ask questions anonymously, remain anonymous to your classmates, or choose to be identified. Using this open forum system the entire class benefits from the instructor and TA responses. In addition, you and other students can also answer each other's questions (again you may choose to remain anonymous or identify yourself to the instructors or everyone). You may still email the instructor or TAs, but more than likely you will be redirected to this message board for help.

Learning Assistant Program (LAP)

This course is supported by the CSE Learning Assistant Program (LAP).
The mission of the LAP is to improve student comprehension and retention in computing fields by focusing on the learner's experience. This course will be supplemented by Learning Assistants (LAs) and Course Leaders (CLs) to help improve your learning. LAs and CLs are other undergraduate students who have taken the same or similar courses and have been trained to help you succeed in this course. Your LAs and CLs will hold regular office hours, help with grading, and assist you with labs/assignments.

I strongly encourage you to utilize the LAs and CLs when you are completing coursework. More information can be found in the Learning Assistant Program Module on Canvas.

Getting Help

Your success in this course is ultimately your responsibility. Your success in this course depends on how well you utilize the opportunities and resources that we provide. There are numerous outlets for learning the material and getting help in this course:

- Lectures: attend lectures regularly and when you do use the time appropriately. Do not distract yourself with social media or other time wasters. Actively take notes (electronic or hand written). It is well-documented that good note taking directly leads to understanding and retention of concepts.
- Lecture Videos: Lecture videos are intended as a supplement that mirrors lecture material but that may not cover everything. Watch them at your own pace on a regular basis for reiteration or in case you missed something in lecture.
- Required Reading: do the required reading on a regular basis. The readings provide additional details and depth that you may not necessarily get directly in lecture.

- Labs & Hack Sessions: use your time during lab and hack sessions wisely. Engage with your lab instructors, teaching assistants, your partner(s) and other students. Be sure to adequately prepare for labs by reading the handouts before coming to lab. Get started and don't get distracted.
- Online message board: if you have questions post to our online message board. It is the best and likely fastest way to get help with your questions. Also, be sure to read other student's posts and questions and feel free to answer yourself!
- Office Hours & Student Resource Center: the instructor and teaching assistants hold regular office hours throughout the week as posted on the course website. Attend office hours if you have questions or want to review material. The Student Resource Center (SRC, https://computing.unl.edu/current-undergraduate#SRC) Monday through Friday. Even if your TAs are not scheduled during that time, there are plenty of other TAs and students present that may be able to help. And, you may be able to help others!
- Don't procrastinate. The biggest reason students fail this course is because they do not give themselves enough opportunities to learn the material. Don't wait to the last minute to start your assignments. Many people wait to the last minute and flood the TAs and SRC, making it difficult to get help as the due date approaches. Don't underestimate how much time your assignment(s) will take and don't wait to the week before hand to get started. Ideally, you should be working on the problems as we are covering them.
- Get help in the *right way*: when you go to the instructor or TA for help, you must demonstrate that you have put forth a good faith effort toward understanding the material. Asking questions that clearly indicate you have failed to read the required material, have not been attending lecture, etc. is *not acceptable*. Don't ask generic questions like "I'm lost, I don't know what I'm doing". Instead, explain what you have tried so far. Explain why you think what you have tried doesn't seem to be working. Then the TA will have an easier time to help you identify misconceptions or problems. This is known as "Rubber Duck Debugging" where in if you try to explain a problem to someone (or, lacking a live person, a rubber duck), then you can usually identify the problem yourself. Or, at the very least, get some insight as to what might be wrong.