Computer Science I

Syllabus

Department of Computer Science & Engineering

University of Nebraska–Lincoln

Fall 2018

,

> "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

1 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

For all other information, see the course website.

2 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.

3 Schedule

See the course website.

4 Relationship of Course to ACE

This course will satisfy Learning Outcome 3: Use computational and formal reasoning (including reasoning based on principles of logic) to solve problems, draw inferences, and determine reasonableness.

Learning Opportunities

The lectures, together with homework and programming assignments and the weekly structured laboratory sessions, teach students methods for developing and implementing algorithms to solve problems. That is, the course not only teaches students about how to design algorithmic solutions, but also teaches students about how to engineer designs into working software. The engineering process of designing and implementing a program involves significant debugging, testing, and refining code. These activities teach and reinforce reasoning and inferencing: a student must develop tests to reasonably indicate program correctness and must draw inferences when diagnosing why a program does not compile, crashes, or generates incorrect output. Also, an algorithm is fundamentally a logical sequence of steps that, given a set of inputs, generates a set of outputs. The course includes approximately:

- 45 hours of lectures each designed to explore concepts and paradigms that are central to the field of computer science.
- (at least) 15 hours of structured laboratory sessions, each designed to train students to apply what they learn in the lectures to actual implementations and analyses of algorithms and software.
- Several homework and programming assignments designed to help students learn about methods for designing algorithmic solutions and the practices of implementing solutions as correct software.

Outcome Assessment

A variety of student work is used to assess achievement of the outcomes, including exams, homework and programming assignments, and structured laboratory work. Exams require students to demonstrate their knowledge in a written format. The programming assignments are inherently practical demonstrations of problem solving and algorithm development, with reasoning and inferencing to produce programs that compile, run, and compute the correct output. The laboratory work supplements the lectures and to provide supervised hands-on experiences of problem solving, algorithm development, and the realization of a computer solution. The students submit their results from solving the lab problems and performing the specified tasks. Laboratory pre-tests, worksheets, and post-tests are graded. The student must pass pre-tests prior to beginning the lab, complete worksheets with results, and take post-tests prior to the end of the lab. In summary, all of the following provide opportunities for students to demonstrate their skills related to the learning objective:

- There are midterm exams and a comprehensive final exam.
- There are several programming assignments.
- There are weekly structured laboratory assignments.

5 Accommodations for Students with Disabilities

It is the policy of the University of Nebraska-Lincoln to provide flexible and individualized accommodations to students with documented disabilities that may affect their ability to fully participate in course activities or to meet course requirements. To receive accommodation services, students must be registered with the Services for Students with Disabilities (SSD) office, 232 Canfield Administration, 472-3787 voice or TTY.

6 Grading

Grading will be based on labs, "hacks", assignments, and exams with the following point distributions.

Category	Number	Points Each	Total
Labs	14	10	140
Hacks	14	25	350
Assignments	5	50	250
Midterm	1	100	100
Final	1	150	150
Total			990

6.1 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 will be made for individual assignments or exams.

6.2 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.

Labs are setup as a *peer programming* experience. In each lab, you will 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

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

immediate pairing, you are encouraged to help and interact and with other pairs in the lab.

Unless otherwise stated, you are required to finish the lab by the end of your regular lab meeting time. A lab instructor must sign off on your lab worksheet and you must turn it in to receive credit. Labs that have not been completed on time may not receive credit.

6.3 Hacks

There will be weekly *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, but it is due by 23:59:59 on Friday in the week in which it is assigned.

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. They are intended to "jumpstart" you on small programming exercises to give you a good start on the full programming assignments.

6.4 Assignments

There will be several programming assignments that you will work outside of class/lab. Carefully read each handout and follow all instructions. Failure to do so may result in points being deducted. Full rubrics are made available through the course website.

6.5 Exams

There will be a midterm exam and a comprehensive final exam. These will be open-book, open-note, *required computer* exams. You will be doing live coding exercises and submitting them online for grading. A working laptop with the proper development tools and resources is required to take these exams. If you do not have a working laptop computer during the exam dates, please contact us for alternative accommodations. More details will be announced closer to the exam dates.

6.6 15th Week Policy Notification

A per UNL's 15th Week Policy (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 and hack will be due during the 15th week or "dead week."

6.7 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).

6.8 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.

Homework assignments have a strict due date/time as defined by the CSE server's system clock. All program files must be handed in using CSE's webhandin as specified in individual assignment handouts. 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.

6.9 Webgrader Policy

Failure to adhere to the requirements of an assignment in such a manner that makes it impossible to grade your program via the webgrader 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 webgrader.

6.10 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 http://cse.unl.edu/academic-integrity

7 Communication & Getting Help

The primary means of communication for this course is Piazza, an online forum system designed for college courses. We have established a Piazza group for this course and you should have received an invitation to join. If you have not, contact the instructor immediately. With Piazza 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 Piazza for help.

In addition, there are two anonymous suggestion boxes that you may use to voice your concerns about any problems in the course if you do not wish to be identified. My personal box is available on the course webpage. The department also maintains an anonymous suggestion box available at https://cse.unl.edu/contact-form.

7.1 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.
- Piazza: if you have questions ask them on Piazza. 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, `http://cse.unl.edu/src`) is open 9AM to 7PM 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.