ECE 2400 Computer Systems Programming, Fall 2021

Course Syllabus

School of Electrical and Computer Engineering
Cornell University

revision: 2021-09-06-21-58

1. Course Information

Cross-Listings	ENGRD 2140 Computer Systems Programming	
Prereqs	CS 1110 (preferred) or CS 1112	
Instructor	Prof. Christopher Batten, 323 Rhodes Hall, `cbatten@cornell.edu` Office Hours: Tuesday, 4:45–5:45pm @ 323 Rhodes Hall Zoom Hours: Tuesday, 7:30–8:30pm	
Graduate TAs	Nick Cebry	Office/Lab Hours: Thu, 7:30–9:30pm @ 225 Upson Hall
	Ryan McMahon	Office/Lab Hours: Thu, 7:30–9:30pm @ 225 Upson Hall
Undergraduate TAs	Guadalupe Bernal	Office/Lab Hours: Mon, 7:30–9:30pm @ 225 Upson Hall
	Michael Egbueze	Office/Lab Hours: Tue, 7:30–9:30pm @ 225 Upson Hall
	Eric Hall	Office/Lab Hours: Wed, 7:30–9:30pm @ 225 Upson Hall
	Sonal Parab	Office/Lab Hours: Wed, 7:30–9:30pm @ 225 Upson Hall
	Anya Prabowo	Office/Lab Hours: Mon, 7:30–9:30pm @ 225 Upson Hall
	Chidera Wokonko	Office/Lab Hours: Tue, 7:30–9:30pm @ 225 Upson Hall
Lectures	Mon/Wed/Fri, 10:10–11:00am @ 219 Phillips Hall	
Disc. Section	Fri, 2:40–3:30pm & 3:45–4:35pm @ 225 Upson Hall	
Required Textbook	zyBook: ECE 2400 Computer Systems Programming Link: `http://learn.zybooks.com` Code: TBD Cost: $88	
Optional Textbook	A. Hilton and A. Bracy, "All of Programming," 2015 Only available as an ebook ($10) `http://aop.cs.cornell.edu`	
Website	`http://www.csl.cornell.edu/courses/ece2400`	
Staff Email	`ece2400-staff-l@cornell.edu`	

2. Description

Computer systems programming involves developing software to connect the low-level computer hardware to high-level, user-facing application software and usually requires careful consideration of performance and resource constraints. Examples of computer systems software include compilers, operating systems, databases, numerical libraries, and embedded controllers. This course aims to provide a strong foundation in the art, principles, and practices of computer systems programming using C and C++, the languages of choice for system-level programmers.

The course lectures are structured into four parts. In the first part, students will use C to explore procedural programming (e.g., functions, conditional statements, iteration statements, recursion, types, pointers, arrays, dynamic allocation). In the second part, students will apply their knowledge of C to explore basic data structures and algorithms (e.g., lists and vectors, complexity analysis, sorting algorithms, abstract data types). In the third part, students will transition from C to C++ and then use C++ to explore four programming paradigms: object-oriented programming (e.g., C++ classes and inheritance for dynamic polymorphism), generic programming (e.g., C++ templates for static polymorphism), functional programming (e.g., C++ functors and lambdas), and concurrent programming (e.g., C++ threads and atomics). In the fourth part, students will apply their knowledge of multi-paradigm C++ to explore more algorithms and data structures involving trees (e.g., binary search trees, binary heaps), tables (e.g., lookup tables, hash tables), and graphs (e.g., depth-first search, breadth-first search, shortest path, minimum spanning trees).

The course includes hands-on discussion sections and a series of five programming assignments for students to put the principles they have learned into practice. Students will gain experience with UNIX software development including command line development tools, distributed version control, unit testing frameworks, continuous integration, debugging tools, and performance evaluation. In the final two programming assignments, students will work in pairs to design, implement, test, and evaluate a high-performance handwriting recognition system to automatically classify handwritten letters.

3. Objectives

This course is meant to be a foundational course in computer systems programming. The course will prepare students for more advanced coursework in computer engineering (e.g., embedded systems, computer architecture) as well as more advanced coursework that focuses on a single type of computer systems software (e.g., compilers, operating systems, databases). By the end of this course, students should be able to:

- **describe** a variety of algorithms and data structures and how to analyze these algorithms and data structures in terms of time and space complexity;
- **apply** the C and C++ programming languages to implement algorithms and data structures using a variety of programming paradigms;
- **evaluate** various algorithm and data structure alternatives and make a compelling qualitative and/or quantitative argument for why one approach is superior;
- **create** non-trivial C/C++ programs (roughly 1,000 lines of code) and the associated testing strategy starting from an English language specification; and
- **write** concise yet comprehensive technical reports that describe a program implemented in C/C++, explain the testing strategy used to verify functionality, and evaluate the program to characterize its performance and memory usage.

4. Prerequisites

This course is targeted towards sophomore-level undergraduate students, although it is also appropriate for advanced freshman students and upperclassman. An introductory course on computing is required. Students need to be comfortable using at least one programming language (e.g., Python through CS 1110 or MATLAB through CS 1112) and should have some experience in software design, development, and testing. No prior knowledge of the C or C++ programming languages is necessary.

5. Topics

The course includes four parts. The first two parts cover procedural programming in C and then use C to explore basic algorithms and data structures, while the second two parts cover multi-paradigm programming in C++ and then use C++ to explore more algorithms and data structures. A list of topics for each part is included below. The exact topics covered in the course are subject to change based on student progress and interest.

- **Part 1: Procedural Programming**
  - Topic 1: Introduction to C (variables, functions, conditional & iteration statements)
  - Topic 2: C Recursion (single vs. multiple recursion)
  - Topic 3: C Types (builtin, user-defined, type checking/inference/conversion/casting)
  - Topic 4: C Pointers (call-by-value/by-pointer, conceptual storage vs machine memory)
  - Topic 5: C Arrays (iterating, arrays as function parameters, strings)
  - Topic 6: C Dynamic Allocation (malloc/free, heap)
- **Part 2: Basic Algorithms and Data Structures**
  - Topic 7: Lists and Vectors (interface, impl, singly/doubly linked list, bounded/resizable vec)
  - Topic 8: Complexity Analysis (time and space complexity)
  - Topic 9: Sorting Algorithms (insertion, selection, merge, quick, hybrid, radix)
  - Topic 10: Abstract Data Types (sequences, stacks, queues, priority queues, sets, maps)
  - Topic 11: Standard C Libraries (stdc, glib)
- **Part 3: Multi-Paradigm Programming**
  - Topic 12: Transition to C++ (namespaces, references, exceptions, new/delete)
  - Topic 13: Object-Oriented Programming (C++ classes & inheritance, dynamic polymorphism)
  - Topic 14: Generic Programming (C++ templates, static polymorphism)
  - Topic 15: Functional Programming (C++ functors and lambdas)
  - Topic 16: Concurrent Programming (C++ threads and atomics)
- **Part 4: More Algorithms and Data Structures**
  - Topic 17: Trees (binary search trees, binary heaps)
  - Topic 18: Tables (lookup tables, hash tables)
  - Topic 19: Graphs (DFS, BFS, shortest path, minimum spanning trees)
  - Topic 20: Standard C++ Libraries (stl, boost)

6. Textbooks

There is both a required textbook from zyBook and an optional textbook which is available as an ebook through the Google Play Store.

6.A Required Textbook

The required textbook for the course is the following custom zyBook assembled specifically for this course by the instructor:

- Title: ECE 2400/5400 Computer Systems Programming
- Link: `http://learn.zybooks.com`
- Code: TBD
- Cost: $88

zyBooks are online, interactive textbooks that makes it easy and engaging to learn principles and then immediately put them out in practice using a series of integrated participation activities, challenge activities, and zyBook labs. The zyBook for this course includes a web-based C/C++ development tool that makes it trivial to try out small code snippets. The zyBook includes one chapter for each of the 20 topics covered in the course. Students can complete the readings either before or after the topics are covered in lecture. **The material included in the course zyBook is a superset of what is covered in lecture. Students are responsible for all material covered in lecture and in the course zyBook.**

The zyBook includes three kinds of interactive activities:

- zyBook Participation Activities: Required, factors into participation grade
- zyBook Challenge Activities: Optional
- zyBook Labs: Required, factors into quiz grade

The first chapter of the zyBook is available free of charge to students after they create a zyBook account. Students can complete zyBook activities in the first chapter while they are finalizing their decision to enroll in the course. Although their work will be saved, this work will not be visible to the instructors until students actually purchase the zyBook. Note that zyBook will offer a full refund for students that drop the course within the first few weeks of the semester. Email `support@zybooks.com` to request a refund.

Students sometimes ask if the zyBook is worth it. Here are some comments from students in past offerings of the course:

- "zyBook is a good supplementary learning material for me to enhance my understanding after a week's lectures."
- "The zybooks were great way to learn the content in short bits and it was extremely helpful. I would not want to take this class without it because it made it a lot easier to get coding practice before doing a PA."
- "I have nothing bad to say about Zybooks. So much better than a paper textbook. Really good opportunities to practice coding too."
- "I really found the zybook helpful. I thought the interactive coding activities helped solidify the information learned in the chapters."
- "zyBook was really helpful because it helped us visualize the subject we were learning by providing diagrams to explain concepts. It also helped reinforce what we were learning by giving mini-quizzes in the middle of the reading."

- "The Zybook helped with giving me background knowledge on topic before we actually learned the topic class. As a result of this, I think I was able to absorb the information a little better the second time it was taught."
- "I thought the zybook was an excellent resource that taught the lecture material in a different way."
- "Definitely a great resource. I found that it was fairly helpful to review material before the exams."
- "I really liked the Zybook. I liked that it was interactive and you actually had to do a little bit of coding. I thought it was fun way to engage with the material. I found it helpful that we also got immediate feedback from our answers (i.e. whether our answer was right/wrong). I don't really have anything negative to say about it; it was much more enjoyable compared to regular textbook."

6.B Optional Textbook

The optional textbook for the course is *"All of Programming,"* by A. Hilton and A. Bracy (2015). This book is completely optional, but can be a great resources for students that need additional background material and/or for students that want to learn about more advanced topics. The book costs $10 and is only available as an ebook through the Google Play Store. The book includes over seven hours of really fantastic embedded videos. To learn more about how to purchase and read the book use this link: `http://www.csl.cornell.edu/courses/ece2400/readings.html`.

7. Format and Procedures

This course includes a combination of lectures, short quizzes, discussion sections, assigned readings, online discussion, programming assignments, and exams.

- **Lectures –** Lectures will be from 10:10am to 11:00am every Monday, Wednesday, and Friday in 219 Phillips Hall excluding the following academic holidays: Labor Day (9/6), Indigenous People's Day (10/11), and Thanksgiving (11/24,11/26). We will start promptly at 10:10am so please arrive on time. Students are expected to attend all lectures, be attentive during lecture, and participate in class discussion. Class participation will be assessed through the participation portion of each student's final grade. Lectures will extensively rely on a set of handouts that students can annotate and fill in throughout the semester. The instructor will provide hard copies of the handouts for the first week of lectures. After the first week of lectures, Students are responsible for either: (1) printing out the handouts ahead of time; (2) purchasing a course packet from the Cornell Bookstore for $22 which includes all of the handouts for the entire semester; or (3) using a digital version of the handouts with a tablet. Please turn off all cellular phones during class. Use of cellular phones and laptops during lecture is not allowed (see Section 11.C). Tablets are allowed.
- **Quizzes –** There will be a zyBook coding quiz (also called zyBook labs) and/or an in-class paper quiz at the beginning of some lectures. zyBook coding quizzes are open-book and must be submitted using the zyBook online system before the start of lecture. In-class paper quizzes are closed-book and should take about five minutes. In-class paper quizzes may or may not be announced ahead of time. There are no make-up zyBook coding or in-class paper quizzes. The lowest two quiz scores are dropped which effectively provides for up to two excused absences. Solutions to in-class paper quizzes will be available online soon after the quiz is given for formative self-assessment.

- **Discussion Section –** The discussion sections will be on Fridays at 2:30–3:20pm or 3:35–4:25pm in 225 Upson Hall. Attendance at the weekly discussion sections is required and attendance will usually be taken. Discussion section participation will be assessed through the participation portion of each student's final grade. These discussion sections will be relatively informal, with the primary focus being on facilitating student's ability to complete the programming assignments and on reviewing material from lecture using problem-based learning.
- **Readings –** Students are expected to complete all of the required reading according to the schedule on the course website. The material included in the course zyBook is a superset of what is covered in lecture. Students are responsible for all material covered in lecture and in the course zyBook. There will be assigned readings in the zyBook, and students will need to complete the participation challenges corresponding to each of these readings. Completing these participation challenges will be assessed through the participation portion of each student's final grade. The zyBook labs serve as coding quizzes and will factor into the quiz portion of each student's file grade.
- **Online Discussion –** Engagement via the Ed online discussion forums is a critical component of the course. Students are required to actively participate on Ed, and this participation will be assessed as part of each student's final grade. There are many ways student can participate including posting questions, liking other questions, responding that they have the same question, helping to create answers, and posting interesting related content.
- **Programming Assignments –** The course includes five programming assignments. Students are expected to work individually on the first three programming assignments and in a group on the final two programming assignments. Students will be using the `ecelinux` servers to complete the programming assignments, the code must be submitted via GitHub, and the report must be submitted in PDF format via the online Canvas assignment submission system (see Section 12). No other means of submission will be accepted. Programming assignments are due on Thursdays at 11:59pm except for the final programming assignment which is due on a Tuesday at 11:59pm (see Section 11.D for late assignment policy). Each programming assignment also includes a milestone which must be submitted one week before the final programming assignment is due.
- **Prelim and Final Exams –** The course includes two prelim exams and a cumulative final exam. The exams assess student understanding of the material presented in lecture and assigned readings. The prelim exams include a series of short answer questions. The final exam include two parts: the first part include a series of short answer questions, while the second part include a detailed design problem which require students to compare and contrast two design alternatives. The exams do *not* assess student understanding of the Linux development environment (e.g., the Linux command line, `make`, `gdb`, etc.) used for the programming assignments. If students have a scheduling conflict with the exam, they must let the instructor know as soon as possible, but no later than two weeks before the prelim or final exam. Graded final exams and the exam solutions are only available for review under the supervision of a course instructor. You may not remove your graded exam, nor may you remove the exam solutions.

8. Assignment and Exam Schedule

The current schedule is on the course website. All programming assignments are due on Thursdays at 11:59pm except for the final programming assignment which is due on a Tuesday at 11:59pm. Changes to this schedule will be posted as announcements via Ed.

Thu	Sep	16	PA1: Incremental Milestone
Thu	Sep	23	PA1: Math Functions
Thu	Sep	30	PA2: Incremental Milestone
Thu	Oct	7	PA2: List and Vector Data Structures
Tue	Oct	19	Prelim from 7:30-9:00pm (location TBD)
Thu	Oct	21	PA3: Incremental Milestone
Thu	Oct	28	PA3: Sorting Algorithms
Thu	Nov	4	PA4: Incremental Milestone
Thu	Nov	11	PA4: Handwriting Recognition System: Linear vs. Binary Search
Thu	Nov	18	PA5: Incremental Milestone
Tue	Nov	23	Prelim from 7:30-9:00pm (location TBD)
Tue	Nov	30	PA5: Handwriting Recognition System: Trees vs. Tables
Tue	Dec	7	PA5: Incremental Milestone
	TBD		Final Exam (location TBD)

9. Grading Scheme

Each part or criteria of every assignment is graded on a four-point scale. A score of 4.25 is an A+, 4 roughly corresponds to an A, 3 roughly corresponds to a B, 2 roughly corresponds to a C, and below a 2 roughly corresponds to C- or lower. A score of 4.0 usually indicates that the submitted work demonstrates no misunderstanding (there may be small mistakes, but these mistakes do not indicate a misunderstanding) or there may be a very small misunderstanding that is vastly outweighed by the demonstrated understanding. A score of 3.0 usually indicates that the submitted work demonstrates more understanding than misunderstanding. A score of 2.0 usually indicates that the submitted work demonstrates more misunderstanding than understanding. A score of 1.0 usually indicates that the submitted work is significantly lacking in some way. A score of 4.25 is reserved for when the submitted work is perfect with absolutely no mistakes or is exceptional in some other way.

Total scores are a weighted average of the scores for each part or criteria. Parts or criteria are usually structured to assess a student's understanding according to four kinds of knowledge: basic recall of previously seen concepts, applying concepts in new situations, qualitatively and quantitatively evaluating alternatives, and creatively implementing new designs; these are ordered in increasing sophistication and thus increasing weight. In almost all cases, scores are awarded for demonstrating understanding and not for effort. Detailed rubrics for all quizzes, programming assignments, and exams are provided once the assignment has been graded to enable students to easily see how the score was awarded.

The final grade is calculated using a weighted average of all assignments. All quiz grades are averaged to form a single total. Students can drop their lowest two quiz scores. Participation will be assessed in part based on: (1) lecture attendance and participation; (2) discussion section attendance and participation; (3) zyBook participation activities; (4) online Ed discussion; and (5) completing student evaluations. The weighting for the various assignments is shown below.

Participation	5%	
Quizzes	5%	(students can drop lowest two scores)
PA Milestones	5%	(weighted equally)
PA Code	20%	(weighted equally)
PA Reports	10%	(weighted equally)
Prelim Exams	30%	(weighted equally)
Final Exam	25%	

Note that the exams account for over half of a student's final grade. The exams in this course are very challenging. Successful students begin preparing for the exams far in advance by carefully reviewing the assigned readings, independently developing study problems, and participating in critical study groups.

To pass the course, a student must at a bare minimum satisfy the following requirements: (1) submit four out of the five programming assignments; (2) take both prelim exams; and (3) take the final exam. **If a student does not satisfy these criteria then that student may fail the course regardless of the student's numerical grade.** The instructor reserves the right to award a D letter grade for students who barely satisfy this criteria but are clearly making no real effort to engage in the course and their own learning.

10. Degree Requirements

This section describes how ECE 2400 can satisfy various degree requirements. Students are responsible for confirming this information with the appropriate student services coordinator.

- **ECE Undergraduate Majors –** We strongly encourage all ECE undergraduate majors to consider taking ECE 2400. ECE 2400 / ENGRD 2140 can be used as your second engineering distribution course (all ECE students must take ECE 2300 as their first engineering distribution course). Alternatively, ECE students may be able to use this course as an outside-ECE technical elective. Students are reminded they are only allowed to use one ECE course as an outside technical elective provided that the course's subject matter lies outside the student's major disciplinary area as determined by the focus of the student's upper-level ECE course work. Regardless, this course satisfies the ECE advanced programming requirement.
- **ECE Undergraduate Minors –** ECE 2400 / ENGRD 2140 can potentially be used as an engineering distribution course (see "Other Undergraduate Students" below) towards your major requirements. However, ECE 2400 does not satisfy any of the specific requirements for the ECE undergraduate minor.
- **CS Undergraduate Majors –** CS majors should take CS 2110 / ENGRD 2110 as their first engineering distribution course. Since both CS 2110 / ENGRD 2110 and ECE 2400 / ENGRD 2140 are part of the same "scientific computing" ENGRD category, you cannot use ECE 2400 as your second engineering distribution course. However, CS majors can still use ECE 2400 as an advisor-approved elective or a major-approved free elective.
- **ECE/CS Undergraduate Double Majors –** A recent change to the CS course requirement means that ECE/CS double majors can now take ECE 2400 / ENGRD 2140 *instead* of CS 2110 / ENGRD 2110. Taking ECE 2400 no longer complicates the path to a double major.
- **CS Undergraduate Minors –** ECE 2400 / ENGRD 2140 can now also be substituted for CS 2110 to satisfy the specific requirements for the CS undergraduate minor. We encourage CS

undergraduate minors wishing to focus on computer systems to consider taking ECE 2400 instead of CS 2110.

- **Other Undergraduate Students –** Students are required to take two ENGRD courses from different categories. ECE 2400 / ENGRD 2140 is part of the "scientific computing" category, so students can use this course as their second engineering distribution course assuming the first is not also in the "scientific computing" category (i.e., not ENGRD 2110, ENGRD 2112, ENGRD 3200). ECE 2400 / ENGRD 2140 is a great way to strengthen your programming skills and can nicely complement core classes in many other disciplines.

11. Policies

This section outlines various policies concerning auditors, usage of cellular phones and laptops in lecture, turning in assignments late, regrading assignments, collaboration, copyright, and accommodations for students with disabilities.

11.A Auditor Policy

Casual listeners that attend lecture but do not enroll as auditors are not allowed; you must enroll officially as an auditor. *If you would like to audit the course please talk to the instructor first!* Usually we wait until the second week of classes before allowing auditors to enroll, to ensure there is sufficient capacity in the lecture room. The requirements for auditors are: (1) attend most of the lectures; (2) complete most of the zyBook coding quizzes and in-class paper quizzes; and (3) perform reasonably well on these quizzes. If you do not plan on attending the lectures and keeping up with the reading for the entire semester, then please do not audit the course. Please note that students are not allowed to audit the course and then take it for credit in a later year unless there is some kind of truly exceptional circumstance.

11.B Course Re-Enrollment Policy

Students are not allowed to enroll for credit for a significant fraction of the course, drop or switch to auditor status, and then re-enroll for credit in a later year. A “significant fraction of the course” means after the first prelim; by this time the student will have: attended several lectures, completed multiple programming assignments, and completed several quizzes. The student should have plenty of experience to decide whether or not they should drop and take the course in a later year. It is not fair for students to have access to assignment solutions and possibly even take both prelims before deciding to drop the course and take it again in a later year; this would essentially enable students to take the course twice to improve their grade.

11.C Cellular Phones and Laptops in Lecture Policy

Students are prohibited from using cellular phones and laptops in lecture unless they receive explicit permission from the instructor. It is not practical to take notes with a laptop for this course. Students will need to write on the handouts, quickly draw state diagrams, and sketch pseudocode during lecture. The distraction caused by a few students using (or misusing) laptops during lecture far outweighs any benefit. Tablets are allowed as long as they are kept flat and used exclusively for note taking. If you feel that you have a strong case for using a laptop during lecture then please speak with the instructor.

11.D Late Assignment Policy

Programming assignment reports must be submitted electronically in PDF format and the code must be submitted electronically via GitHub. **No other formats will be accepted!** Programming assignments must be submitted by 11:59pm on the due date unless otherwise specified. No late submissions will be accepted and no extensions will be granted except for a family or medical emergency. We will be using the online Canvas assignment submission system. You can continue to resubmit your files as many times as you would like up until the deadline, so please feel free to upload early and often. **If you submit an assignment even one minute past the deadline, then the assignment will be marked as late.**

As an exception to this rule, each student has five slip-days that may be used when submitting programming assignments (both incremental milestones and the final assignment) throughout the semester. Each slip-day provides an automatic 24-hour extension. **The PA code and report are treated as a single assignment for the purposes of slip days (i.e., a single slip day provides an automatic 24-hour extension for submitting a student's code, report, or both).** When working with a partner, both students must have a slip day to secure an automatic 24-hour extension. Regardless, the maximum automatic extension is 48 hours. The purpose of the slip-day system is to give students the freedom to more effectively manage their time. The due dates for the course are available at the beginning of the semester, so please plan ahead so you can handle weeks with many other deadlines. To use a slip day, simply submit the report late; Canvas will allow assignments to be uploaded up to two days late. You are responsible for keeping track of how many slip days you have remaining. If you accidentally submit an assignment late without the proper number of slip days remaining then, although the system will allow the upload, we will not grade the assignment (or we will grade the latest upload before the due date).

11.E Regrade Policy

Addition errors in the total score are always applicable for regrades. Regrades concerning the actual solution should be rare and are only permitted when there is a significant error. Please only make regrade requests when the case is strong and a significant number of points are at stake. Regrade requests should be submitted online via a private post on Ed within one week of when an assignment is returned to the student or within one week of when an exam is reviewed with the class. You must provide a justification for the regrade request.

11.F Collaboration Policy

The work you submit in this course is expected to be the result of your individual effort only, or in the case of the final programming assignment, the result of you and your group's effort only. Your work should accurately demonstrate your understanding of the material. The use of a computer in no way modifies the standards of academic integrity expected under the University Code.

You are encouraged to study together and to discuss information and concepts covered in lecture with other students. You can give "consulting" help to or receive "consulting" help from other students. Students can also freely discuss basic computing skills or the course infrastructure. **However, this permissible cooperation should never involve one student (or group) having possession of or observing in detail a copy of all or part of work done by someone else, in the form of an email, an email attachment file, a flash drive, a hard copy, or on a computer screen.** Students are not allowed to seek consulting help from online forums outside of Cornell University. Students are not allowed to use online solutions (e.g., from Course Hero) from previous offerings of this course. Students are encouraged to seek consulting help from their peers and from the course staff via office hours and

the online Ed discussion forums. **If a student receives consulting help from anyone outside of the course staff, then the student must acknowledge this help on the submitted assignment.**

During zyBook coding quizzes, in-class paper quizzes, and examinations, you must do your own work. Talking or discussion is not permitted during the in-class paper quizzes and examinations, nor may you compare papers, copy from others, or collaborate in any way. Students must not discuss a quiz/exam's contents with other students who have not taken the quiz/exam. If prior to taking it, you are inadvertently exposed to material in an quiz/exam (by whatever means) you must immediately inform an instructor.

Should a violation of the code of academic integrity occur, then a primary hearing will be held. See `https://theuniversityfaculty.cornell.edu/academic-integrity` for more information about academic integrity proceedings.

Examples of acceptable collaboration:

- Bob is struggling on a zyBook coding quiz about object oriented programming, so he seeks consulting help from Alice, a fellow student in the course. Alice goes through various examples from the lecture and reading materials to help Bob understand the concepts, and they sketch a few state diagrams related to different problems (*not* the coding quiz problem) on a whiteboard. Bob and Alice work independently on the zyBook coding quiz. At no time do Bob and Alice actually collaborate on the zyBook coding quiz so there is no need for Bob to acknowledge the consulting help he received from Alice.
- Ben is struggling to complete a programming assignment which requires implementing a linked list. He talks with Alice and Cathy and learns that all three students are really struggling. So the three students get together for a brainstorming session. They review the lecture and reading materials and then sketch on a whiteboard some ideas on how to implement a linked list. They might also sketch out some code snippets to try and understand the best way to implement the data structure. Then each student independently writes the code for the assignment and includes an acknowledgment of the help they received from the other students. At no time do the students actually share code.
- Alice and Amy are having trouble figuring out difficult test cases for their handwriting recognition system. They post on Ed to see if anyone has some general ideas for tricky corner cases. Ben and Bob figured out an interesting test case that ensures their handwriting recognition system correctly handles the worst case input, so Ben and Bob post a qualitative description of this test case. Alice and Amy independently write the code for this test case and then include an acknowledgment of the help they received from the other group. At no time do the groups actually share test code.

Examples of unacceptable collaboration:

- Bob is struggling on a zyBook coding quiz about object oriented programming, so he seeks consulting help from Alice, a fellow student in the course. *Alice shows Bob her completed zyBook code and walks him through the various steps required to solve the quiz.* Bob takes some notes during their discussion, and then independently writes his own code. Bob acknowledges the help he received from Alice as a comment in his zyBook coding quiz, but it doesn't matter since Alice explicitly shared her code with Bob.
- Anna cannot make today's lecture since she has an extracurricular commitment. Anna asks Bart to complete two in-class paper quizzes and to put Anna's name on the second quiz. This misrepresents Anna's attendance and is not allowed.

- Ben is struggling to complete a programming assignment which requires implementing a linked list. He talks with Alice and Cathy and learns that all three students are really struggling. So the three students get together for a joint coding session. Each student works on one method of the linked list class, and then they combine these methods together to create the final working linked list class. *The three students share and copy each other's code often in order to finish the assignment.* Each student submits the final code independently. Each student acknowledges the help he or she received from the other students, but it doesn't matter since they explicitly shared code.
- Alice and Amy are having difficulty figuring out difficult test cases for their handwriting recognition system. They post on Ed to see if anyone has some general ideas for tricky corner cases. Ben and Bob figured out an interesting test case that ensures their handwriting recognition system correctly handles the worst case input, so *Alice and Amy send their test code to Ben and Bob via email.* Alice and Amy modify this test code and then include it in their submission. Alice and Amy include an acknowledgment of the help they received from the other group, but it doesn't matter since they explicitly shared code.

Notice that **the key is that students should not share the actual solutions or code with each other unless expressly permitted by the course instructors.** Consulting with your fellow students is fine and is an important part of succeeding in this course.

11.G Copyright Policy

All course materials produced by the course instructor (including all handouts, tutorials, homeworks, quizzes, exams, videos, scripts, and code) are copyright of the course instructor unless otherwise noted. Download and use of these materials are permitted for individual educational non-commercial purposes only. Redistribution either in part or in whole via both commercial (e.g., Course Hero) or non-commercial (e.g., public website) requires written permission of the copyright holder.

11.H Accommodations for Students with Disabilities

In compliance with the Cornell University policy and equal access laws, the instructor is available to discuss appropriate academic accommodations that may be required for students with disabilities. Requests for academic accommodations are to be made during the first three weeks of the semester, except for unusual circumstances, so arrangements can be made. Students are encouraged to register with Student Disability Services to verify their eligibility for appropriate accommodations.

12. Online and Computing Resources

We will be making use of a variety of online websites and computing resources.

- **Public Course Website –** `http://www.csl.cornell.edu/courses/ece2400` is the main public course website which will also have course details, updated schedule, reading assignments, and most handouts. We intend for all course content to always be available on Canvas. The public course website is just for public access to some of this content.
- **Canvas Course Site –** We will be using Canvas to manage course content, assignment submission, and grade distribution.
- **Ed –** We will be using Ed for all announcements and discussion on course content, lab assignments, and the projects. The course staff is notified whenever anyone posts on the forum and will respond quickly. Using the forum allows other students to contribute to the discussion and to see the answers. Use common sense when posting questions such that you do not reveal so-

lutions. Please prefer posting to Ed as opposed to directly emailing the course staff unless you need to discuss a personal issue.

- `ecelinux` **Servers –** The ECE department has a cluster of Linux-based servers which we will be using for the programming assignments. You can access the `ecelinux` servers remotely using PowerShell, Mac Terminal, VS Code, X2Go, MobaXterm, or Mac Terminal with X11. More information about accessing the ECE computing resources is available on the Canvas course site.
- **GitHub –** GitHub is an online Git repository hosting service. We will be using the commerical GitHub service to distribute programming assignment harnesses and as a mechanism for student collaboration on the final two programming assignments. Students will also use GitHub for submitting the code for their programming assignments. Students are expected to become familiar with the Git version control system. Note that we are not using the Cornell hosted version of GitHub as in some other courses; we are using `github.com`.
- **Codecov.io –** Codevoc.io is an online code-coverage visualization service that is tightly coupled to GitHub. Codecov.io will automatically display code-coverage reports after each GitHub commit. We will be using the results reported by Codecov.io to help evaluate the verification quality of the programming assignments.