DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
UNIVERSITY OF TENNESSEE, KNOXVILLE

COSC 202 Data Structures
Spring 2026 Syllabus (Draft)

Time and Location: Tuesday & Thursday 12:55 pm-2:10 pm, Min Kao Engineering 622

Instructor:

- Dr. Scott Emrich
  - Office: 608 Min Kao; 974-3891; semrich@utk.edu;
  - Tentative office hours: Mon 2:00pm-3:00pm, after class and by appointment

An Invitation

It is my goal for you to learn simple data structures (then more complex ones in the fall), get better at both problem solving and coding, and along the way enjoy this course! I would like to balance us having a positive experience with developing real, applied skills that will be highly useful to you in interviews, this summer, and beyond. Remember the TAs and I are here at every step to help. To support your learning, we will have both active learning experiences and a dedicated lab period on Wednesday.

Course Webpage: https://web.eecs.utk.edu/~semrich/cs202-26/

Short Course Description

This course is part one of our two course sequence that focuses on fundamental data structures and associated algorithms. During the semester we will cover basic data structures with a focus on I/O, files, pointers, and basic memory management, lists, queues, stacks, hash tables and an introduction to algorithm analysis. For these topics students are expected to design and implement code with the overall goal of becoming more self-sufficient (C++) programmers. All programming will take place in a Unix environment and will be reinforced by weekly assignments. We will emphasize **conceptual understanding and use**, not full general implementations of all structures covered.

Textbook

C++ How to Program by Deitel and Deitel (optional);

Course outcomes

By the end of this first course in this sequence, you will be able to competently design C++ objects/classes, with information hiding, and understand and use more advanced

data structures and methods including but not limited to lists, maps, stacks, queues, and trees. Specifically, you will be able to:

1. Demonstrate understanding of, and proficiency in use of C++/object-oriented concepts including data hiding, templates, and common object-oriented design practices. Assessed via exam and programming assignments.
2. Analyze the performance of data structures in order to select the right one for each situation, as well as create or extend data structures to fit new situations. Assessed in exams and in programming assignments.
3. Debug pointer-based code using systematic reasoning, tests, and memory tools.
4. Combine data structures to solve real world problems, employing abstractions to make them work together cleanly and safely. Assessed through the final lab.

Major Topics:

1. Review of C++/programming basics inc. strings and OOP (5 hours)
2. Iterators, polymorphism, exceptions (3 hours)
3. Pointers, lists and deques (4 hours)
4. Hashing (3 hours)
5. Stacks, queues, doubly linked list data structures (2 hours)
6. Sets and maps (2 hours)
7. Recursion (3 hours)
8. Running times and Big-Oh (3 hours)
9. Basics of trees: binary search trees and AVL trees (5 hours)
10. Priority queues and binary heaps (2 hours)
11. Applied Extensions / Integration Topics (7 hours)
12. Collaborative reviews and midterm (2.5 hours)

Grading: Assignments will be penalized 10% per day, prorated. As other courses I teach, this semester you can submit one updated assignment for re-grading by the end of the semester, which will be used for computing your final grade.

Final grades will be computed from a weighted sum of points as follows, although I will probably take informal feedback from you on the weighting:

65%: homework (including programming and written answers submitted)
10%: midterm exam
15%: final exam
10%: class participation

Course percentages will be translated into letter grades as follows: A: 95% and up; A-: 92-95%; B+: 88-92%; B: 85-88%; B-: 82-85%; C+: 78-82%; C: 75-78%; C-: 72-75%; D: 65-72%; D-: 62-65; F: 0-62%. Absences will only be excused in accordance with University policy.

Extensions and "Things Happen" Policy

This course is designed with built-in flexibility through partial credit where possible, a transparent late penalty, and a resubmission opportunity (aka the "redo") at the end of

the semester. These mechanisms are intended to handle the most common disruptions that arise during a busy term.

Therefore requests for deadline extensions due to workload management, competing deadlines, illness, unexpected events or starting late will generally not be approved, as these situations are already accounted for by the late policy and resubmission option.

Final lab: A slightly larger lab/project may be due at the end of the semester. For now, these will be solo projects, but we can discuss whether you can also work in pairs.

ADA statement: If you need an accommodation based on a disability can contact Dr. Emrich privately. Full accommodation will be made once approved.

Academic Code of Honor: Any instance of academic dishonesty will not be tolerated. All graded work should go from your head to your fingers to submission; no copying of complete solutions or code that solves a key topic being assessed (group or online).

Like your non-EECS courses, you are encouraged to use "resources" to help solve non-critical elements, e.g., you forget how to convert a string to a number (int) in a lab focused on linked lists. You can google and find info on a site like GeeksforGeeks but you ** must ** cite the URL. You can also ask a generative AI/co-pilot, but you must cite in your comments what you used, and it is your responsibility the solution works as intended (aka passes tests). If in doubt, ask me or the TAs. As a concrete example, if the lab is about linked lists, you may not use AI to generate a full linked-list implementation — but you may ask about syntax, debugging, or language features as long as you cite the AI tool and explain the usage briefly in your comments.

You are encouraged to discuss high level solutions with peers but to help us, please note who you spoke with about what in the comments of your submitted solution.

Attendance and Time management:

No matter how you look at it, programming is a time-intensive activity that is best done throughout the week (and not right before it is due). To be fair to all, all regular labs will be due before the next lab period. Please upload partial work to Canvas and we will provide partial credit if we can.

Even if we partially move to a hybrid mode, students will still be expected to attend and contribute regularly. In addition to participating in discussions, some challenges will be provided to reinforce course content and provide a platform to start working with other programmers.

<u>Class participation will be assessed as follows:</u>

0.5 points – Show up at lab on time (once per week)
1 point –Participate in a lab by having a "meaningful" conversation with a TA that indicates that you are making progress (see below; once per week.)
1 point – Stop by Dr. Emrich's "virtual" office hours this semester (max one point)

0.5 points – Participate in a classroom exercise or follow up (max seven points.)

During the pandemic, I shrunk labs from 3 hours down to roughly 1 hour and added a "prep" video. That ended up working very well and the videos were one of the most beloved elements of the two last go-rounds of 202. We will reinforce the video with effectively a forced office hour that you will get easy participation credit for attending. We really want to have a "I've gotten this far and my code does this; I've set it up in this way" conversation, and to reinforce that your labs will be due on Wednesday. Please do feel welcome to ask us for explanation about the instructions, though; we're happy to.

Your class participation grade will be calculated as (points earned / 21 estimated), so you can earn a few extra credit points by actively participating in lectures and pass/fall challenges throughout the semester. Because the instructor realizes that sometimes "things happen" you will be able to earn more points than you need. This usually is done as a "bonus" such that a large percentage of the class has 100% based on the attendance pattern of that semester. More on the first day of class.

How to Succeed in this Course (TL;DR)

Students who succeed typically:

- Start labs early
- Attend and participate in our in-class exercises
- Talk through partial solutions with TAs when unsure
- Use your resubmission strategically
- Treat debugging as part of learning, not failure

We'll give more pointers throughout the first week of class.