

Detailed Syllabus for CSS 343

Data Structures, Algorithms, and Discrete Math II

This core computer science course covers the internal workings of algorithms and data structures, from mathematical principles to implementation in C++. Topics include development of algorithms; algorithm analysis; object-oriented programming; abstract data types including trees, priority queues, graphs, and tables; regular expressions and context-free grammars.

The objective of the course is to refine and extend the concepts and practical skills introduced in CSS 342. By the end of this quarter, you will be a confident C++ programmer and will be comfortable with the basics of object-oriented design and programming. You will understand how to analyze a problem and design a solution, recognizing when existing techniques and software are reusable. You will understand the tradeoffs among memory, running time, and implementation time associated with different data structures and algorithms.

The subject matter is highly technical so plan to put in considerable time and effort master the material. Expect to spend an average of 15 hours a week outside of class time for this course; some of you may spend more, some less time.

Course Objectives

The goals of this course are for you to be able to do the following:

- Be able to apply pictorial representations to explain the implementations of the data structures and their operations: Tree, Binary Search Tree (BST), Balanced BST, Heap, Graph, Hash Table.
- Be able to understand and implement algorithms associated with data structures: Binary Search Tree (BST), Heap, Graph, Hash Table.
- Be able to implement data structures of common data abstractions.
- Be able to write regular expressions and context-free grammars to generate languages.
- Be able to design and produce code using object-oriented programming and appropriate design patterns.

ABET Student Outcomes

For engineering accreditation purposes, this course supports the following ABET student outcomes:

- Outcome (a): An ability to apply knowledge of mathematics, science, and engineering. (Strongly Supported)

- Outcome (b): An ability to design and conduct experiments, as well as to analyze and interpret data. (Supported)
- Outcome (c): An ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability. (Minimally Supported)
- Outcome (e): An ability to identify, formulate, and solve engineering problems. (Strongly Supported)
- Outcome (k): An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice. (Supported)

Prerequisites

You should have at least a minimum grade of 2.0 in CSS 301, a minimum grade of 2.0 in CSS 342, and a minimum grade of 2.0 in either STMATH 125 or MATH 125.

Instructor

Michael Stiber <stiber@uw.edu (<mailto:stiber@uw.edu>)>, room UW1-360D, office phone (425) 352-5280 (I almost never answer). Office hours Mondays 1-2 and Wednesdays 2:30-3:30 in the Linux lab or by appointment (for example, if you'd like to meet privately).

Textbooks

Frank M. Carrano and Timothy Henry, *Data Abstraction & Problem Solving with C++: Walls and Mirrors*, sixth edition, Pearson, 2013.

Charles A. Cusack and David A. Santos, *An Active Introduction to Discrete Mathematics and Algorithms*, version 2.5.1, December 21, 2015. Available as a free online PDF [here](http://www.cs.hope.edu/~cusack/Notes/Notes/Books/Active%20Introduction%20to%20Discrete%20Mathematics%20and%20Algorithms/ActiveIntroToDiscreteMathAndAlgorithms.2.5.1.pdf) (<http://www.cs.hope.edu/~cusack/Notes/Notes/Books/Active%20Introduction%20to%20Discrete%20Mathematics%20and%20Algorithms/ActiveIntroToDiscreteMathAndAlgorithms.2.5.1.pdf>).

Suggested Readings

Bruce Eckel, *Thinking in C++*, Second Edition, vols. 1 & 2, Prentice Hall, 2000 and 2003. A good “from scratch” introduction to the language. Available for free in electronic form on-line [here](http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html) (<http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html>).

Bjarne Stroustrup, *The C++ Programming Language*, Third Edition, Addison-Wesley, 2000. The canonical reference. Make sure you get the third edition.

Harley Hahn, *Harley Hahn's Student Guide to UNIX*, 2nd edition, McGraw-Hill, 1996. If you're unfamiliar

with UNIX or LINUX and want to learn more, get a book like this.

Herbert Schildt, *STL Programming from the Ground Up*, Osborne/McGraw-Hill, 1999. A very good introduction to the Standard Template Library, with lots of examples. However, it is not a reference; for example, it doesn't provide complete lists of methods for each class.

Nicolai M. Josuttis, *The C++ Standard Library: A Tutorial and Reference*, Addison-Wesley, 2012. Much more of a complete reference than the Schildt book, though it has been updated to include more tutorials and examples.

Homework

We will have both written exercises and programming assignments. While all programming assignments will have a value of 100 points, the value of written exercises will vary (likely in the range of 15–30 points). Subsequent sections of this syllabus carefully spell out (in detail) both the procedures for program submission and the content of what you should submit. *Please read this syllabus and the [CSS 343 Design and Coding Standards \(https://canvas.uw.edu/courses/1112451/pages/css-343-design-and-coding-standards\)](https://canvas.uw.edu/courses/1112451/pages/css-343-design-and-coding-standards) page carefully.* If there is anything you don't understand or are not sure about *ask me.* *I will assume that you have done so, and will mark off if what you submit does not match what is required.*

Programming Assignments

The philosophy behind the programming assignments is to exercise your growing abilities to solve problems using computers. Note that I do *not* say to exercise your programming ability; I assume that, though you may be a beginner, you are basically familiar and comfortable with the process of writing and debugging software. In this class, you will learn about a variety of problem-solving tools: algorithms and data structures (taken together, abstract data types), approaches (for example, inheritance), and mathematical techniques to compare and develop new algorithms (Boolean algebra, finite state machines, grammars, etc). The homeworks are explicitly designed to be substantial — they will require you to use what you learn in a systematic manner. As an example, imagine that we have just covered the topic of binary search trees (BSTs), and in class discussed a BST of integers. I would not assign a homework that has you implement a BST of strings. The point of learning about BSTs is to understand them so that you can use them to solve problems (and to know when they are appropriate for use). Therefore, a more suitable assignment would be one in which you are asked to implement a simple database, which internally might use a BST to index primary keys. This would allow you to investigate how a generic abstract data type's capabilities can be related to the specific functionality of a particular program.

Programming assignments will be due at specific dates and times. I will *not* accept any lateness in this class — if your assignment is submitted even a few minutes late, it will not be graded, and you will

receive a *zero* for that assignment. In fact, we will be using the UW [Collect It](https://catalyst.uw.edu/collectit/dropbox/stiber/39323) (<https://catalyst.uw.edu/collectit/dropbox/stiber/39323>) system (not Canvas), which will not accept late submissions. Except for special circumstances, such as medical and other emergencies, no exceptions will be made to this policy (this includes crashed/eaten/lost disks — *make frequent backups*). You are more than welcome to submit work before the due date.

Programming Assignment Procedures. We will be dividing assignments into two phases, and you will be responsible for completing each phase by the indicated deadline. that you carefully read and follow the procedures in this syllabus. I will present an initial assignment description during class and will expect you to commence on the assignment immediately. *I make these assignment purposefully vague and incomplete in certain respects.* The two phases are “specification and design” and “implementation”; they are graded separately — the former as part of a peer design review and the latter in testing by me.

1. It will be your task during the initial specification and design phase (see “Specification and Design,” later) to define the problem to be solved and produce a solution. This will involve identifying where the original assignment description is incomplete, and at the next class meeting, asking questions to clarify matters so you can finalize the specification. Your specification and design will be due *one week after the assignment is handed out*. We will break into groups for a [peer design review](https://canvas.uw.edu/courses/1112451/pages/peer-design-review-process) (<https://canvas.uw.edu/courses/1112451/pages/peer-design-review-process>), we will discuss the assignment and design issues together as a class, and then you will hand in your designs so you may receive credit. Your name and student number should be written on your hard copy submissions. Please strive to either write/draw clearly or use a computer and high-quality printer to prepare your documentation; I cannot give you credit for what I cannot read. Note that this represents one-half of the time allotted to your programming assignment. Therefore, before handing in your preliminary specification and design, you should ask yourself, “Does this represent 50% of the work I’ll do on this program?”
2. At this point, you will have roughly a week to implement your design. You will submit your program using the UW [Collect It](https://catalyst.uw.edu/collectit/dropbox/stiber/39323) (<https://catalyst.uw.edu/collectit/dropbox/stiber/39323>) system. The [Collect It](https://catalyst.uw.edu/collectit/dropbox/stiber/39323) (<https://catalyst.uw.edu/collectit/dropbox/stiber/39323>) area will only accept tar or zip files, and the submission size is strictly limited, so please submit only source code. I will unpack your submission into a directory dedicated to your assignment, compile all of the .cpp files into a single executable, and test it. Testing will be performed using g++ on one of the CSS Linux machines. *It is your responsibility to ensure:*
 - That, when unpacked, all of your files will be present, in a the same directory as the archive (i.e., the archive should only contain files, not a subdirectory), and have appropriate capitalization (Unix is case-sensitive; Windows is inconsistent regarding case sensitivity. If I need to manually rename all your files, I will reduce your program grade). I suggest that you test to make sure that your submission workflow is correct by transferring your archive file to one of the CSS Linux machines, unpacking it, checking that all of the files are present in the same directory (and not a

subdirectory), and then compiling it and running it.

- That any long lines in your software are neither truncated nor wrapped.
- That your program will compile using g++ on a CSS Linux machine using g++. The course web site and CSS wiki has information about lab 320, Linux, and Unix. g++ is our course compiler; if you choose to use another compiler, then be forewarned that you have made a decision that may increase the time it takes for you to complete the assignment (as you may need to modify your code to get it working under g++). If you make that decision, please be responsible for the outcome — do not expect that I will take into consideration that it compiled and ran under another compiler.
- That the input your program expects and the output it produces *exactly* matches the specifications in the assignment.
- That your name and student number are in a comment in each source file.
- That you follow all other coding conventions outlined below.

I allocate credit based on your coding style, your documentation, and on your program's execution characteristics ("correctness", determined by *black-box* testing). For example, if your program does not compile under g++, you will receive *zero* points for correctness. I will run your program against a set of test cases (which I will *not* release ahead of time); partial credit will be awarded if it passes only some of them. I will *not* debug your program or try to figure out why it doesn't work — partial credit only comes from passing test cases. Any other way of assigning partial credit would, in my opinion, be unfair: based more on my debugging ability than the qualities of the program. Because of this, I require you to design your program before you write code, and I strongly urge you to implement your program in stages, so that you always have a partially working version. If you use a development environment other than g++, then I suggest that you periodically move your code to a machine that has g++ to compile and test it (again, I do not endorse the approach of using another compiler, I am merely suggesting prudent practice). Of course, I am more than happy to meet with you about your program before or after the due date.

Program Grading. Generally speaking, adherence to coding standards will be worth 15% and program correctness (in other words, does it work?) 85%. However, depending on the specific nature of each assignment, the exact percentages (and any other aspects' weights) may change. However, I will inform you if that is the case.

Tests

Tests will be in-class, closed book, with one page (one side of a sheet of letter-sized paper) of notes allowed. There will be a midterm and a (cumulative) final.

Grading

Your course average is computed as: 35% homework + 10% designs/peer design reviews + 25% midterm + 30% final

I don't grade on a curve. I compute everyone's quarter average based on the formula above. I then use my judgment to determine what averages correspond to an 'A', 'B', etc. for the quarter. Some quarters' assignments, etc. turn out harder, and so the averages are lower. Other quarters, averages are higher. I adjust for that at the end. Decimal grades are then computed using the equivalences in the Time Schedule, linearly interpolating between letter-grade boundaries. Furthermore, I am well aware of the significance of assigning a grade below 2.0, in terms of impact on your career here at UWB. I can assure you that I examine in detail the performance in this course of each student before assigning a grade below 2.0.

What is the difference between this and grading on a curve? With the latter, the goal is to have X% 'A's, Y % 'B's, etc. My way, I would be happy to give out all 'A's (if they were earned). FYI, in a "typical" quarter, below 65% might be a 'D', 65%–75% a 'C', 75%–85% a 'B', and above 85% an 'A'. You may use this as a rough guide; however, if you really want to know how you're doing, please see me. *I reserve the right to adjust these scores to reflect the specifics of assignments, test questions, etc. for each quarter.*

Access and Accommodations

Your experience in this class is important to me. If you have already established accommodations with Disability Resources for Students (DRS), please communicate your approved accommodations to me at your earliest convenience so we can discuss your needs in this course.

If you have not yet established services through DRS, but have a temporary health condition or permanent disability that requires accommodations (conditions include but not limited to; mental health, attention-related, learning, vision, hearing, physical or health impacts), you are welcome to contact DRS at 425-352-5307 or [uwbdrs@uw.edu \(mailto:uwbdrs@uw.edu\)](mailto:uwbdrs@uw.edu). DRS offers resources and coordinates reasonable accommodations for students with disabilities and/or temporary health conditions.

Reasonable accommodations are established through an interactive process between you, your instructor(s), and DRS. It is the policy and practice of the University of Washington to create inclusive and accessible learning environments consistent with federal and state law.

For Our Veterans

If you are a student who has served in our nation's military forces, *thank you* for your service. I hope that you feel comfortable enough to confidentially self-identify yourself to me so I can help you make a successful transition from the military to higher education.

Problems

TOPICS

If you have problems with anything in the course, please come and see me during office hours, make an appointment to see me at some other time, or send email. I want to make you a success in this course. If you have trouble with the assignments, see me before they are due. If you fall behind, it will be difficult to catch up. I will not give out grades of "incomplete" except in extreme circumstances.

Etiquette, Etc. (the "rules")

- Read this entire detailed syllabus, the [design and coding standards for this class](https://canvas.uw.edu/courses/1112451/pages/css-343-design-and-coding-standards) (<https://canvas.uw.edu/courses/1112451/pages/css-343-design-and-coding-standards>), and the other class documents listed under the [Pages](https://canvas.uw.edu/courses/1112451/pages) (<https://canvas.uw.edu/courses/1112451/pages>) link to the left. *Every word*. Failure to follow instructions (for example, neglecting coding or documentation standards, trying to turn an assignment in the wrong way or after a deadline) will adversely impact your grade.
- Please read the readings for each class meeting before that meeting. Come to class prepared to ask questions to clarify anything you are unsure about.
- I will not try force you to use any particular development environment. However, your programs *must* compile with g++ and run on the CSS Linux machines. Please be advised that, if you use any other environment, it is possible that you will spend considerable extra time "porting" your code to the class compiler and computing environment. It is even possible that you will never get your program working. It is a worthwhile investment of your time for you to learn to use Unix and g++. If you use another development environment, then you assume all responsibility for getting your code running on the class computing platform; I will *not* make exceptions and test programs otherwise.
- Similarly, there are certain class standards for documentation, including UML diagrams. If the tool you use does not produce diagrams that look like the class requirements, please find another tool. Neat hand drawings are perfectly acceptable.
- You are responsible for making back-up copies of your work. Disk crashes, etc. are *not* acceptable reasons for extensions of assignment due dates. Note that your Linux and UW IT home directories are professionally backed up, as are Google Drive, Microsoft OneDrive, etc.
- Assignments are due when specified. Barring illness or similar extenuating circumstances, please do not attempt to submit amendments, bug fixes, or forgotten material after the fact.
- While I do not formally require your attendance in class, participation in peer design reviews is a portion of your grade. Additionally, I will assume that you are cognizant of everything that is covered in class, including clarifications of programming assignments, changes in due dates, etc. Material covered in class is fair game for assignments and tests, regardless of its absence from the textbooks.
- Please use your UW email (*netid@uw.edu*) for communications with me (i.e., make sure that that shows as your return address). This is the only way that I can authenticate that you are who you say you are. I will endeavor to only email you at that address. I do this because that address is the only one guaranteed to be connected to you, based on information in the student database.

- I expect you to treat fellow students, and other UWB community members, with the utmost respect and consideration.
- Please be considerate of other students regarding your activities in class. Do not arrive late, leave early, get up at random times, walk about, leave the classroom and return, etc. I will make sure we take a break halfway through each class; you make sure that you take care of whatever personal business you need to so that you can make it to the break. I understand that this class is scheduled around dinnertime. If you do bring food, please ensure that you don't generate noise or smell. Finally, please do not use a computer during class for anything other than note taking or compiling and testing code.
- As far as exams are concerned, I hope I don't need to remind anyone that it's not acceptable to take a break in the middle — i.e., leave the room for any reason — and expect to be able to resume taking the exam afterwards. It's not a matter of me doubting your honesty — it's a matter of you, as a professional, avoiding doing anything that might even hint at being less than honest and focused on the task at hand. Please ensure that you've either taken care of any personal matters before the exam or have provided me with documentation of a medical condition that precludes spending two hours at uninterrupted work.
- You are expected to provide original work based on your own effort for this course. You will receive a zero for any coursework for which you are discovered cheating or plagiarizing. You will be referred to the University for further action. It is your responsibility to know and uphold the Student Conduct Code for the University of Washington, available at <http://www.uwb.edu/students/policies/> (<http://www.uwb.edu/students/policies/>).

CSS 343 A Wi 17: Data Structures, Algorithms, And Discrete Mathematics II

[Jump to Today](#)

 [Edit](#)

This is a summary and schedule for CSS 343; the full syllabus is located [here](#).




Course Summary















This core computer science course covers the internal workings of algorithms and data structures, from mathematical principles to implementation in C++. Topics include development of algorithms; algorithm analysis; object-oriented programming; abstract data types including trees, priority queues, graphs, and tables; regular expressions and context-free grammars.














The objective of the course is to refine and extend the concepts and practical skills introduced in CSS 342. By the end of this quarter, you will be a confident C++ programmer and will be comfortable with the basics of object-oriented design and programming. You will understand how to analyze a problem and design a solution, recognizing when existing techniques and software are reusable. You will understand the tradeoffs among memory, running time, and implementation time associated with different data structures and algorithms.

The subject matter is highly technical so plan to put in considerable time and effort master the material. Expect to spend an average of 15 hours a week outside of class time for this course; some of you may spend more, some less time.

Course Summary:

Date	Details	
Wed Jan 4, 2017	 Course Introduction; Trees (https://canvas.uw.edu/calendar?event_id=959664&include_contexts=course_1112451)	5:45pm to 7:45pm
Mon Jan 9, 2017	 Tree Implementation and Applications (https://canvas.uw.edu/calendar?event_id=959665&include_contexts=course_1112451)	5:45pm to 7:45pm
Wed Jan 11, 2017	 Binary Search Trees; Priority Queues (https://canvas.uw.edu/calendar?event_id=959666&include_contexts=course_1112451)	5:45pm to 7:45pm

	 Program 1: BSTs (design and peer review) (https://canvas.uw.edu/courses/1112451/assignments/3509372)	due by 5:45pm
Mon Jan 16, 2017	 No Class; MLK Day (https://canvas.uw.edu/calendar?event_id=959669&include_contexts=course_1112451)	12am
Wed Jan 18, 2017	 Heaps (https://canvas.uw.edu/calendar?event_id=959667&include_contexts=course_1112451)	5:45pm to 7:45pm
	 Program 1: BSTs (program) (https://canvas.uw.edu/courses/1112451/assignments/3509380)	due by 11:59pm
Mon Jan 23, 2017	 Graphs and their implementation (https://canvas.uw.edu/calendar?event_id=959668&include_contexts=course_1112451)	5:45pm to 7:45pm
Wed Jan 25, 2017	 Graph Algorithms (https://canvas.uw.edu/calendar?event_id=959671&include_contexts=course_1112451)	5:45pm to 7:45pm
	 Program 2: Graphs (design and peer review) (https://canvas.uw.edu/courses/1112451/assignments/3509612)	due by 5:45pm
Mon Jan 30, 2017	 Balanced Search Trees (https://canvas.uw.edu/calendar?event_id=959673&include_contexts=course_1112451)	5:45pm to 7:45pm
Wed Feb 1, 2017	 Balanced Search Trees, cont'd (https://canvas.uw.edu/calendar?event_id=960641&include_contexts=course_1112451)	5:45pm to 7:45pm
	 Program 2: Graphs (program) (https://canvas.uw.edu/courses/1112451/assignments/3509615)	due by 11:59pm
Mon Feb 6, 2017	 No class due to snow (https://canvas.uw.edu/calendar?event_id=986766&include_contexts=course_1112451)	5:45pm to 7:45pm
Wed Feb 8, 2017	 Dictionaries and Hashing; Midterm review (https://canvas.uw.edu/calendar?event_id=959679&include_contexts=course_1112451)	5:45pm to 7:45pm
	 Program 3: Hash Tables (design and peer review) (https://canvas.uw.edu/courses/1112451/assignments/3509618)	due by 5:45pm
Mon Feb 13, 2017	 Midterm Exam (https://canvas.uw.edu/calendar?event_id=959685&include_contexts=course_1112451)	5:45pm to 7:45pm

	 Midterm (https://canvas.uw.edu/courses/1112451/assignments/3509771)	due by 7:45pm
Wed Feb 15, 2017	 C++ Programming Idioms (https://canvas.uw.edu/calendar?event_id=959689&include_contexts=course_1112451)	5:45pm to 7:45pm
Mon Feb 20, 2017	 No class; Presidents Day (https://canvas.uw.edu/calendar?event_id=959686&include_contexts=course_1112451)	12am
Wed Feb 22, 2017	 Object-Oriented Design Techniques (https://canvas.uw.edu/calendar?event_id=959691&include_contexts=course_1112451)	5:45pm to 7:45pm
	 Program 3: Hash Tables (program and report) (https://canvas.uw.edu/courses/1112451/assignments/3509619)	due by 11:59pm
Mon Feb 27, 2017	 Finite State Machines (https://canvas.uw.edu/calendar?event_id=959684&include_contexts=course_1112451)	5:45pm to 7:45pm
Wed Mar 1, 2017	 Regular Expressions (https://canvas.uw.edu/calendar?event_id=959687&include_contexts=course_1112451)	5:45pm to 7:45pm
	 Program 4: O-O Techniques (design and peer review) (https://canvas.uw.edu/courses/1112451/assignments/3509623)	due by 5:45pm
Mon Mar 6, 2017	 Context Free Grammars (https://canvas.uw.edu/calendar?event_id=959702&include_contexts=course_1112451)	5:45pm to 7:45pm
Wed Mar 8, 2017	 Parsing (https://canvas.uw.edu/calendar?event_id=959703&include_contexts=course_1112451)	5:45pm to 7:45pm
	 Program 4: O-O Techniques (program) (https://canvas.uw.edu/courses/1112451/assignments/3509670)	due by 11:59pm
Mon Mar 13, 2017	 Final Exam (https://canvas.uw.edu/calendar?event_id=959778&include_contexts=course_1112451)	5:45pm to 7:45pm
	 Final (https://canvas.uw.edu/courses/1112451/assignments/3509773)	due by 7:45pm