CSS 343-A Data Structures and Algorithms

Winter 2014

Time and Location:	3:30–5:30 p.m. M/W, UW1 020
Instructor:	Mike Stark
Prerequisites:	CSS 342 (with minimum grade of 2.0)
Web Page:	`http://courses.washington.edu/css343/stark`[‡]
Textbook:	*Data Abstraction and Problem Solving with C++* Frank M. Carrano Addison-Wesley, ISBN 0132830310

Course Description: This sequenced course integrates mathematical principles with detailed instruction in computer programming. 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.

Activities and Goals

Refining and extending the concepts and skills introduced in CSS 342, students develop competencies associated with problem-solving, design, testing, and programming techniques. Topics include ADTs, data structures, related algorithms, and object-oriented design and programming. Formal automata theory as it applies to programming languages is introduced. Good software engineering and algorithm analysis techniques are used throughout. As with most technical courses, besides ability and motivation, it takes time to learn and master the subject. Expect to spend some 15 hours a week or more outside of class time for this course.

The course involves completion of a combination of homework exercises, programming projects, and examinations. Some smaller programming assignments will be classified as homework; major programming projects may require the submission of accompanying design documents. It is strongly suggested that each student adopt a sensible schedule for working on programs as soon as they are assigned, because the documentation alone involves a significant amount of work.

Homework Exercises Homework exercises will be assigned throughout the term. Although there may be some hard-copy assignments to be turned in at the beginning of class, most will be submitted electronically. Instructions will be provided.

Programming Assignments Programming assignments, in which students are expected to implement constructs and apply techniques presented in class, will be assigned regularly. The details may vary with the assignment, but in most cases students will be expected to develop C++ programs that have a specified function or structure. Programs will be graded on both correct operation as well as proper formatting and style of the course code. The source code must be submitted for grading.

- Your code must compile and run properly using the linux g++ compiler.
- Your work will be graded on documentation (clarity and completeness), style (indentation and use of blank lines/spaces), meaningful identifier names, organization of your program (modularity/design), efficiency (no useless, unnecessary, or unnecessarily complicated code), output (clarity and format), the overall readability, and following directions. Coding/documentation style guidelines and a detailed grading rubric can be found linked from the course website.

[‡] To request academic accommodations due to a disability, please contact Disability Resources (DRS) at 425.352.5307, 425.352.5303 TDD, or drs@uwb.edu. Provide documentation by the third week of the quarter.

- Expect to receive little or no credit for submitted code that does not compile and run. Run-time errors with not much output yield a low grade. Run-time errors or incorrect answers will result in a significant number of points being deducted from your grade. If your code does not compile and run, the program will receive little feedback. If you do not put in the time to write it, time will not be put in to grade it.

Programming assignments and electronically submitted homework assignments are to be submitted via Catalyst by the specified due date. Keep in mind that catalyst can be slow, so do not wait for the last minute to submit. Unless we have spoken about the circumstances and prior arrangements have been made, an assignment not turned in receives a grade of zero.

Exams There will be two in-class written examinations: a midterm and a final.

Policies

Collaboration and Academic Honesty With the exception of specific group projects, each student must work independently on homework and programs. While course participants are allowed (and encouraged) to discuss the problem statement and general principles, assist each other with compiler issues and debugging, design and coding must be done individually. Each student's work is expected to be original; use of unauthorized external resources is not allowed. Examples of such include published implementations of algorithms and code developed by other students in other courses—including previous offerings of this course. To avoid being an unwitting participant in dishonest activity, keep your work to yourself. Be careful not to leave printouts in the printer or laying around.

This class is run by an honor code. By taking this class, you agree that you will not collaborate inappropriately on any work and your submitted work will be original. Violation of this betrays the trust of the instructor and the university, and will be taken seriously.

In-Class Etiquette Computer use during lecture is limited to taking notes and possibly testing small pieces of code or looking up lecture-related information. Visiting social networking sites, dealing with email, playing games, watching video, etc., are all inappropriate during class. Please silence cell phones, and keep whispering to a minimum.

The class period is two hours long, which allows for a ten minute break approximately in the middle of class. Keep in mind that the instructor needs the break too, so please keep questions to a minimum; in particular there is unlikely to be enough time to debug any code during the break.

Message Boards Message boards will on the Catalyst system will be available. Students are encouraged to post and respond as well as read the message boards regularly. It is, however, inappropriate to post extensive amounts of your source code.

See http://www.uwb.edu/academic/policies/Academic_Conduct.xhtml

Topics and Tentative Schedule

Week	Date	Topic	Reading	Assignments
1	Jan. 6	Preliminaries, Tree introduction		
	Jan. 8	Huffman encoding, Binary Search Tree	C++ 10 (5th), C++ 15, 16 (6th) Math 11.1–11.3	
2	Jan. 13	Priority Queues, Binary Heaps	C++ 11.2 (5th), C++ 13.3, 17 (6th)	
	Jan. 15	Priority Queues, Binary Heaps continued		Hw/lab 1 due
3	Jan. 20	*Holiday—Martin Luther King (no school)*		
	Jan. 22	Graphs (Dijkstra Shortest Path)	C++ 13 (partial, 5th), C++ 20 (partial, 6th), Math 10.1–10.3, 10.6	
	Jan. 24			Hw/lab 2, part 1 due
4	Jan. 27	Graphs cont. (Depth/Breadth-First)		Hw/lab 2, part 2 due
	Jan. 29	Balanced Trees (AVL, 2-3, B-tree)	C++ 12.1, 14.3 (partial, 5th), C++ 19.1, 19.2.1–2.3, 19.5, 21.3.3 (6th)	
5	Feb. 3	Balanced Trees cont., Retrieval (trie)		
	Feb. 5	Object-oriented Design/Programming	C++ 1.1–1.2 (5th, 6th), 1.3 (6th)	Hw/lab 3 due
6	Feb. 10	**Midterm exam**		
	Feb. 12	Inheritance/Polymorphism, Hash Table introduction	C++ 8.1 (5th), interludes (6th) C++ 12.2–12.3 (5th), C++ 18.4 (6th)	
7	Feb. 17	*Holiday Presidents Day (no school)*		
	Feb. 19	Hash Factory, Hash Tables		Sample code
8	Feb. 24	Hash Tables cont., Design Review		Hw/lab4 design due
	Feb. 26	Inheritance/Polymorphism under the hood	C++ 8.2 (5th), interludes (6th)	
9	Mar. 3	Languages, Regular expressions	Math 13.4	
	Mar. 5	Finite Automata	Math 13.3	
10	Mar. 10	Context-free Grammars	Math 13.1, C++ 5.2	
	Mar. 12	Turing Machines, Last day stuff	Course notes	Hw/lab 4 due (implementation)
11	Mar. 17	**Final exam (in class)**		

Grades and Grading Policies

The final grade for this course will be computed using a weighted arithmetic average of homework and programming assignments as well as exams, according to the following:

$$\begin{array}{ll} \text{Homework (programs and exercises):} & 40\% \\ \text{Midterm Exam:} & 30\% \\ \text{Final Exam:} & 30\% \end{array} \qquad \longrightarrow \qquad \begin{array}{ll} \mathbf{4.0}, & \text{if } 97 \le \textit{average}; \\ \mathbf{3.5}, & \text{if } 90 \le \textit{average} < 97; \\ \mathbf{2.7}, & \text{if } 80 \le \textit{average} < 90; \\ \mathbf{1.7}, & \text{if } 70 \le \textit{average} < 80; \\ \mathbf{0.7}, & \text{if } 60 \le \textit{average} < 70; \end{array}$$

Attaining an average in the specified interval guarantees at least that grade. Generally the grade will be interpolated in the interval; e.g., an average of 85% would earn the student a 3.1, which is midway between the 2.7 and 3.5 grades at the boundaries of the 80–90 interval. However, the interpolation is approximate. The interval boundaries may be lowered somewhat at the instructors discretion.

Instructor Contact Information

Office:	Truly House	**Email:**	`mmstark@u.washington.edu`
		Web:	`http://faculty.washington.edu/mmstark`
Office Hours:	*to be announced*		