CSS 342 - Mathematical Principles of Computing (Winter 2012)

Instructor: Valentin Razmov (vrazmov [at] uwb [dot] edu)
Course web: http://courses.washington.edu/css342/vrazmov/
Class meets: MW 5:45pm-7:45pm @ UW1-221
Office hours: MW 5:00pm-5:30pm, 7:50pm-8:20pm @ UW1-360E (or by appointment)
Office: UW1-360E (only during office hours)

Course description:
Along with CSS 343, this fast-paced course is intended to bring you up to speed for taking Junior and Senior level CSS courses. It does this by integrating fundamental mathematics of computing with detailed instruction in program design. By the end of the quarter, you will be familiar with the basics of object-oriented programming, as reflected in much of the C++ language. You will understand how to analyze a problem, design and evaluate a solution. You will know many basic data structures, algorithms, and the tradeoffs among memory usage, running time, and implementation time associated with each. Good software engineering techniques will be used throughout.
Topics include: recursion, computational complexity and algorithm analysis, logic, mathematical proofs and induction, lists, stacks, queues, sorting and searching, data abstraction, and object-oriented methods.
In short, completing the course successfully you will be on your way to becoming a professional in the field. You will also know more clearly some aspects of what a programming job involves, and if it is the right opportunity for you personally.

Textbooks:
(1) *Data Abstraction & Problem Solving with C++ (5th ed.)*, Frank M. Carrano, Addison-Wesley.
(2) *Discrete Mathematics and Its Applications (7th ed.)*, Kenneth Rosen, McGraw-Hill.
(3) A C++ book/tutorial of your choice (*optional, but recommended*)

Some recommended C++ books:
• *Thinking in C++*, Bruce Eckel, Prentice Hall. (An excellent guide for learners and professionals alike; a wonderful resource explaining, unlike others, why things are the way they are; versions of this book are freely available online, too)
• *The C++ Programming Language (3rd ed.)*, Bjarne Stroustrup, Addison-Wesley. (A definitive reference by C++'s author)
• *C++ Primer (3rd ed.)*, Stanley Lippman and Josée Lajoie, Addison Wesley. (A thorough examination of the C++ standard)
• *Effective C++ (2nd ed.): 50 Specific Ways to Improve Your Programs and Designs*, Scott Meyers, Addison-Wesley. (Very well written, examines common shortfalls and proposes strategies for improving code; a must-have for professionals.)
• *More Effective C++: 35 New Ways to Improve Your Programs & Designs*, Scott Meyers, Addison-Wesley. (Another gem from the author of "Effective C++…")
• *C++: How to Program*, H.M. Deitel and P.J. Deitel, Addison-Wesley. (An accessible description)
• *C++ in Plain English*, Brian Overland, Wiley.

Assignments:
There will be four types of assignments:
(a) Written homework assignments of discrete mathematics problems will help you practice your understanding of material related to the mathematical underpinnings of programming, and the analysis of algorithmic solutions. Some of these will be for practice-only, while others will be graded.
(b) Programming projects (a.k.a. labs), gradually increasing in complexity, will give you hands-on experience of object-oriented concepts and related design and programming techniques, via the use of the C++ language.
(c) Short in-class quizzes will be designed to reward you for staying on top of the latest discussed material, including knowledge of terminology.
(d) Although not via a formal assignment, you will be rewarded (see Grading Guideline) for meaningfully contributing to an online discussion forum specific to our class.

Specifics on each assignment will be given as necessary in follow-up discussions and write-ups, including on the course web.

Be sure to *precisely* follow the directions in each assignment (part of your assignment grade is for this). Precision is something that computers demand from us, and programming is to a large extent a work that must pass computer muster (and human expectations, too).

Project Quality Expectations:

You can use any standard C++ compiler and platform, and any IDE to develop your programming code.
However, since Linux is traditionally the platform used to test and grade CSS 342 / CSS 343 programming projects, your code must compile without errors by using g++ under Linux, and must also run properly.
While simple C++ code tends to easily port across platforms, hope is not a plan: you are highly encouraged to set aside reasonable time to test your code on Linux before you submit it, *even if* it works fine on your other platform of choice.
Syntax errors and run-time errors with little output will yield a low score. Run-time errors or incorrect answers will lead to a significant loss of points from your score. In other words, budget time for testing and

TEST YOUR CODE THOROUGHLY!

Project grades will also reflect the quality of: associated documentation (in its clarity and completeness); coding style (indentation, 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); overall readability; and closely following directions in the assignment specification.

Programming Project Tips:

(a) Plan on a few hours for the task of porting your code to Linux, *just in case*. Experience shows that sometimes you may not need it, but sometimes you inevitably will – and we don't know when.
(b) Frequently upload and compile / run / test incremental versions of your code on Linux, especially as submission time approaches.
(c) Keep older recent versions on separate media that *you* control (e.g., a flash drive), so that you can easily revert back if you somehow lose your recent work or make undesirable changes to code in the last minute.

Assignment Submissions and Late / Make-up Policy:

Assignments are to be submitted electronically (and, in some cases, on hard copy, too) to UW Catalyst. A link will be available on the course web when the time approaches.

Due time for electronic submissions will be by 10pm on specified dates, honoring each student's need for proper night's rest.
Submissions late by up to 1 hour will incur a 10% penalty; submissions after 1 hour will not be accepted.
Assignments (or portions thereof) due on hard copy will be accepted no later than the start of class on the given day.
Make-up exams or quizzes will be given only under extenuating circumstances.

Collaboration Policy:

You are explicitly encouraged to work with each other in the following situations:
(a) understanding lecture material and related readings (e.g., via discussions on the online discussion forum);
(b) clarifying problem statements on assignments – *after* each participant has given independent thought to options;
(c) discussing potential design alternatives and testing choices – *after* each participant has written their own candidate design and a set of test cases;
(d) helping to debug programming code – *after* you have made a good faith effort (at least 15 minutes of focused work) to resolve a given problem by yourself.
Actual submitted material – solutions to homework problems, project design descriptions, write-up of test cases, any programming code and associated documentation – must be written individually by you.
The only exception is where group work (e.g., pair programming) is explicitly allowed on projects – typically on more complex projects late in the quarter. I will make a special announcement when this time comes.
If at any point you feel uncertain about whether collaboration in a particular situation is allowed, please ask me.

Tentative Grading Guideline:

The contribution of various graded course aspects to your overall course grade will be approximately this:

Written assignments and quizzes:	15%	
Programming projects & write-ups:	40%	(Note: Later projects will weigh a bit more than earlier ones)
Midterm:	15%	
Final:	20%	
Participation online:	10%	

For an approximation of your course grade, consider the following scale (though not strictly followed):

<60%: 'D' (0.7-1.4) 60%–75%: 'C' (1.5-2.4) 75%–85%: 'B' (2.5-3.4) >85%: 'A' (3.5-4.0)

Other Requirements:
Any other specific requirements will be described along with relevant assignments, and announced on the course web.

Workload Expectations and Qualities to Succeed:
As with most technically intensive courses, to master the subject it takes intrinsic motivation, but also persistence, strong focus and dedication, attention to detail, a proactive attitude, and a healthy dose of curiosity about learning.
Expect to spend an additional 15 hours per week *on average* (or more, if you have done less programming lately) outside of class – most of that time will go to project-related work, including finding relevant resources for problems you may encounter. With time you will rapidly get more efficient and more effective, and you will enjoy that!

Topics Covered and Tentative, Preliminary CSS 342 Schedule:
Below is an approximate ordering of topics. The schedule may evolve. Not all sections and chapters may be covered thoroughly.

Date	Topics	Reading	Assignment
1/04	Welcome; What to expect from the course; C++ language structure & basic OOP; Development tools, Linux	Carrano, Ch. 1.1 (pp.3-9), Ch. 3 (pp.143-150), App. A.1-A.8, A.11-A.12	Lab #1 assigned
1/09	Pointers in C++ (start); Simple sorting algorithms; Recursion (start)	Carrano, Ch. 9.2 + Ch. 1.2 + Ch. 2.1-2.2 + Ch. 3 (pp.154-160) + Ch. 4.1	Lab #1 design review (in class)
1/11	Recursion (cont.); Recurrence relations & induction proofs; Recursion as a problem solving technique	Carrano, Ch. 2.3-2.5 + Ch. 5 + App. D; Rosen, Ch. 5	Written HW #1 assigned
1/16	**Holiday (Martin Luther King, Jr. Day)**		Lab #1 due; Lab #2 assigned
1/18	Data abstraction and OOP in C++ (cont.); Software Engineering Principles; Pointers in C++ (cont.)	Carrano, Ch. 3 + Ch. 1.3 + App. E	Written HW #1 due
1/23	Linked list implementation & memory management	Carrano, Ch. 4	Lab #2 peer design review
1/25	Stacks: implementation and applications	Carrano, Ch. 6	
1/30	Queues: implementation and applications	Carrano, Ch. 7	Lab #2 due; Lab #3 assigned
2/01	C++ objects, classes, and OO design	Carrano, Ch. 8	
2/06	Algorithm analysis	Carrano, 9.1; Rosen, Ch. 3	Lab #3 peer design review; Written HW #2 assigned
2/08	**Midterm exam**		
2/13	Sorting (cont'd)	Carrano, 9.2	Lab #3 due; Lab #4 assigned; Written HW #2 due
2/15	Propositional & predicate logic	Rosen, Ch. 1	Written HW #3 assigned
2/20	**Holiday (President's Day)**		Lab #4 peer design review
2/22	Logic (cont'd)	Rosen, Ch. 1	
2/27	Trees	Carrano, Ch. 10	Lab #4 due; Lab #5 assigned;
2/29	Trees (cont'd)	Carrano, Ch. 10	Written HW #3 due
3/05	Mathematical foundations	Rosen, Ch. 2	Lab #5 peer design review; Written HW #4 assigned (practice)
3/07	Mathematical foundations (cont'd)	Rosen, Ch. 2	Lab #5 due (Friday, 3/09)
3/12	**Final exam**		