EE 599: Computing Principles for Electrical Engineers
Units: 2
Term—Day—Time: Spring 2021, MW 6:30pm to 8:20pm

Midterm Exam: Wednesday March 10th, 7-9 p.m
Final Exam: Wednesday, May 5th, 7-9 p.m

Location: TBD

Instructor: Arash Saifhashemi
Office: EEB 504B
Office Hours: *TBD*
Contact Info: saifhash@usc.edu

Teaching Assistant:
Office: TBD
Office Hours: TBD
Contact Info: TBD

Sept.18.2020

Course Description
This course provides a broad introduction to concepts required for advanced computer programming. It is targeted to graduate electrical engineering students with either informal or only basic programming experience. It teaches knowledge and develops practices required to understand and implement sophisticated software solutions. The course consists of two main parts: (1) foundations of software engineering, and (2) algorithms and data structures. The course includes a theoretical basis to critically analyze algorithms, data structures, and mathematical methods. It emphasizes learning by implementation and priotorizes coding. All assignments and projects include a major programming component to develop correct execution over suggestive pseudo-code. Also, the course emphasizes on modern software engineering methodologies such as source control and testing using programming languages such as C++ and Python.

Learning Objectives
A student that successfully completes this course will:

- Understand code to execution including building, tokenization, compiling, and linking.
- Apply step-wise debugging to identify causes of software defects.
- Utilize modern programming practices such as STL objects and object oriented (OO) programming including inheritance, overloading, templates, and polymorphism.
- Possess knowledge to identify and explain standard algorithms (sort, search, recursion, dynamic programming, backtracking) and data structures (map, set, heap, tree, hash, lists, graphs).
- Develop skills to analyze novel code for its performance in both time and space complexity.
- Be comfortable deciding between multiple solutions given optimization or cost criteria.
- Apply test-driven software design and understand its role in minimizing errors and regression.
- Understand the connection between running software and underlying hardware including basics of threading, interrupts, memory management, caching, and device access.

Prerequisite(s): None
Co-Requisite(s): None
Concurrent Enrollment: None
Recommended Preparation: Exposure to computer programming

Technological Proficiency and Hardware/Software Required
You need access to a full stack for both C++ and Python development. You may consider installing a linux virtual machine to ensure maximum interoperability and access to any tools. The instructor and teaching assistants will give guidance during the first weeks of class.

Required Readings and Supplementary Materials

1. Algorithm, 4th Edition (required)

Robert Sedgewick, Kevin Wayne (available at the campus store)

2. Algorithms in C++, 3rd Edition (optional, C++ supplement to 1)

Robert Sedgewick, Kevin Wayne (available at the campus store)

3. The C++ Programming Language, 4th Edition (recommended)

Bjarne Stroustrup (available at the campus store)

4. Code Complete: A Practical Handbook of Software Construction, 2nd Edition (recommended)

Steve McConnell (available at the campus store)

Note: The texts are secondary to in-class lecture material and homework sets.

Description and Assessment of Assignments
There will be about eight homework assignments. Homework will be assigned every 1-2 weeks. The assignments involve coding implementation in C++ with a proof of correctness and unit tests.

All projects and assignments must be submitted electronically either through source code management (e.g. Github) or for auto-grading. There is no "paper-copy"submission required or allowed. Review requirements for each assignment before submission.

No submission should be "headless" and should include a README file to describe any methods, testing, and validation.

Late submission policy

The following policy and late submission penalties will be applied for homework and project submissions:

- 1 day late: -15 points (of 100)
- 2 days late: -30 points (of 100)
- 3 days late: -45 points (of 100)
- More than 3 days: 0 points

In case of medical or other emergency, you should inform the instructor as soon as possible.

Exams

Ensure that you can stream live video (mute or audio off) during the entirety of the work time for proctoring. You must make prior arrangements with me if that is not possible. You must show how you arrived at your answers to receive full credit. Any cheating may result in an "F" in the course and will be referred to Student Affairs for other penalties. Make up exams will only be given for valid medical or family emergency excuses (proof required).

Extra Credit

Exams and assignment may have extra credit. For example a homework assignment may have 130 points where 100 points is considered as extra credit. The extra credit for each assignment will only apply to that assignment and will not be used as compensation to other assignments or exams.

Attendance and Participation

Attendance is mandatory to all lectures and discussions. You are responsible for missed announcements and changes to the course schedule and assignments. Your attendance may be synchronous or asynchronous. Make synchronous attendance a priority. Per university guidance: you should plan to attend every synchronous session of this class regardless of when it occurs in your time zone. Some unreasonable hours (sessions outside 07:00 – 22:00) may preclude this general rule.

Synchronous class dynamics are improved substantially with visible participants in the class. Arrange to have your cameras on (default: "camera on, audio off") during synchronous online sessions. You must make prior arrangements with me if that is not possible.

USC policy requires that all classes conducted online be recorded for asynchronous viewing with transcriptions made available. These recordings are considered "educational records" subject to federal privacy laws (FERPA) as students may be personally identifiable in class recordings by voice, name, or image. Students are not permitted to create their own class recordings without prior written permission. Violations of these policies will be met with the appropriate disciplinary sanction.

Grading Breakdown

Assessment Tool (assignments)	% of Grade
Homework	35%
Project	25%
Exam #1	20%
Exam #2	20%

TOTAL	

Letter Grades
The following table for your grades:

A	95-100
A-	90-94
B+	87-89
B	83-86
B-	80-82
C+	77-79
C	73-76
C-	70-72
D+	67-69
D	63-66
D-	60-62
F	59 and below

Grading Scale
Depending on the class performance, the grades may or may not be scaled.

Assignment Submission Policy
TBD based on course timeline and instructor.

Grading Timeline
Grading will be handled by both automatic and manual means. Code submissions may be reviewed and applied a series of test cases to determine function.

Course Schedule: A Weekly Breakdown

	Lecture	Readings/Preparation	Homework (Topics from Lecture and Discussion of the same week)

Week 1	Motivation, Introduction, Goals Example: design, algorithm, implementation, analysis Build systems, Unit tests and Version control, Monte Carlo Simulation	**Lecture slides**	HW 1 is assigned: tools and setup
Week 2	Big-O notation Logic (propositional, truth tables, first order, predicate), methods of proof C++ introduction: classes, member variables and methods. Public/private.	**Lecture slides** **[3]: Pt. 1, Sec. 6, 9, 12, 13** [1]: Ch. 1.3, 1.4	HW 2 is assigned: monte carlo simulation (HW 1 is due)
Week 3	C++: Functions, Pointers and references C++ constructor, copy constructor, and destructor. Functors and Lambda functions	Lecture slides [3]: Sec. 7, 8, 15, 16, 17	HW 3 is assigned: functions and pointers (HW 2 is due)
Week 4	Data Structures: array, stack, linked list, queue, tree C++ STL (string, list, vector, set, map, algorithm)	Lecture slides [3]: Ch. 14, 18-21, 27, Pt. 4	HW 4 is assigned: STL (HW 3 is due)
Week 5	Data Structures: heap, tree and traversal algorithms, graph traversal (DFS, BFS, topological) Greedy algorithms: MST, shortest path	Lecture slides [1]: Ch. 3.1 – 3.4	HW 5 is assigned: Graph and Tree traversal (HW 4 is due)
Week 6	Binary search, and binary search trees Recursion, Divide and conquer	Lecture slides [1]: Ch. 2.1 – 2.4	HW 6 is assigned: Algorithms: binary search and BST (HW 5 is due)
Week 7	Sorting algorithms 1	Lecture slides [1]: Ch. 2.1 – 2.4	HW 6 is assigned:
Week 8	Sorting algorithms 2 Benchmarking	Lecture slides [1]: Ch. 2.1 – 2.4	
Week 9	Review and midterm preparation		(HW 6 is due)

	Project discussion		
Week 10	Dynamic programming and memoization	Lecture slides	HW 7 is assigned
Week 11	More on dynamic promming (shortest path algorithms) Backtracking	Lecture slides	(HW 7 due) HW 8 is assigned
Week 12	Multithreading: introduction	Lecture slides	(Project topic due) (HW 8 due)
Week 13	Multithreading: memory management, atomic variables, synchronization and applications Cache and memory, hardware and interrupts, Map reduce	Lecture slides	(project phase 1 due)
Week 14	Advanced C++ concepts: polymorphism, inheritance, move sematics, templates, variadic templates		Project videos and Project phase 2 due
Week 15	Prospects and review		Project Final Report due
Final Exam	Final exam on Wednesday, May 6th, 7-9 p.m		

Projects

Teams of two students (teams of one or three with instructor approval) design and develop a software solution to a self-identified industry or research problem. Teams are encouraged to devise problems of particular interest to their backgrounds, interest, or research. The instructor will guide teams with difficulty identifying a suitable topic. All projects must obtain the instructor's written approval. Teams will prepare and present their *approved* project and show how it applies course material, concepts, and best-practices. Attendance *and participation* during the project presentation session(s) are mandatory.

Requirements

Project topics must include sufficient mathematical and algorithmic complexity and either include or extend substantive material from the course. Teams should treat the project as a platform to demonstrate mastery of design specification, algorithmic analysis, testing, debugging, and result validation. All projects must use the C++ language as the primary computer language unless approved explicitly in writing by the instructor. But teams may use or integrate additional languages for tooling and supporting.

Example projects

1. ***Document database engine***: Develop an interface for writing and retrieving documents from a managed storage engine. The system should not wrap another database engine but should expose a protocol for authentication, communication, and error handling. It may also explore disaster or corruption recovery.

2. ***Semantic translator***: Build a system convert an input string or stream from one defined semantic/language to another. The mapping should not be a trivial one-to-one and should require the retention of state or combination of non-contiguous information.

3. ***Digitial signal processor***: Design a complete software package to load and manipulate digital signal data (e.g. audio, image, video). It should be a complete user experience and provide facility beyond a single-purpose tool. It should include state information to provide feedback based on a sequence of user input such as Undo, Redo, and real-time updating.

Grading and Milestones

Phase 1 – Design, components, classes, and tests	week 13	20%
Phase 2 – Integration and deployment	week 15	25%
Demo and presentation	finals	20%
Project report and video		35%

Deliverables and demo

1. Written project report: the project report should summarize the topic, provide relevant background (theoretical or applied), timeline and contributions, and document challenges and extensions. It should provide discussion sufficient that an uninformed expert could understand the logic, algorithmic decisions, and implementations. Teams should provide quantifiable metrics to justify engineering tradeoffs.
2. Presentation: Approximately 10 minute (depends on class size) presentation to describe to the class their topic problem and their solution. It should provide only what is necessary to understand the "what" and "why" and include minimal theoretical background.
3. Video: 3-4 minute video that describes the problem, your design, and implementation. You may choose to upload this to a video sharing site such as YouTube but that is not required. All team members must participate equally.
4. Source code: submitted to instructor by providing link to pull from github.

Statement on Academic Conduct and Support Systems

Academic Conduct:

Plagiarism – presenting someone else's ideas as your own, either verbatim or recast in your own words – is a serious academic offense with serious consequences. Please familiarize yourself with the discussion of plagiarism in SCampus in Part B, Section 11, "Behavior Violating University Standards" policy.usc.edu/scampus-part-b. Other forms of academic dishonesty are equally unacceptable. See additional information in SCampus and university policies on scientific misconduct, policy.usc.edu/scientific-misconduct.

Support Systems:

Counseling and Mental Health - (213) 740-9355 – 24/7 on call
studenthealth.usc.edu/counseling
Free and confidential mental health treatment for students, including short-term psychotherapy, group counseling, stress fitness workshops, and crisis intervention.

National Suicide Prevention Lifeline - 1 (800) 273-8255 – 24/7 on call
suicidepreventionlifeline.org
Free and confidential emotional support to people in suicidal crisis or emotional distress 24 hours a day, 7 days a week.

Relationship and Sexual Violence Prevention Services (RSVP) - (213) 740-9355(WELL), press "0" after hours – 24/7 on call
studenthealth.usc.edu/sexual-assault
Free and confidential therapy services, workshops, and training for situations related to gender-based harm.

Office of Equity and Diversity (OED) - (213) 740-5086 | Title IX – (213) 821-8298
equity.usc.edu, titleix.usc.edu
Information about how to get help or help someone affected by harassment or discrimination, rights of protected classes, reporting options, and additional resources for students, faculty, staff, visitors, and applicants.

Reporting Incidents of Bias or Harassment - (213) 740-5086 or (213) 821-8298
usc-advocate.symplicity.com/care_report
Avenue to report incidents of bias, hate crimes, and microaggressions to the Office of Equity and Diversity |Title IX for appropriate investigation, supportive measures, and response.

The Office of Disability Services and Programs - (213) 740-0776
dsp.usc.edu
Support and accommodations for students with disabilities. Services include assistance in providing readers/notetakers/interpreters, special accommodations for test taking needs, assistance with architectural barriers, assistive technology, and support for individual needs.

USC Campus Support and Intervention - (213) 821-4710
campussupport.usc.edu
Assists students and families in resolving complex personal, financial, and academic issues adversely affecting their success as a student.

Diversity at USC - (213) 740-2101
diversity.usc.edu

Information on events, programs and training, the Provost's Diversity and Inclusion Council, Diversity Liaisons for each academic school, chronology, participation, and various resources for students.

USC Emergency - UPC: (213) 740-4321, HSC: (323) 442-1000 – 24/7 on call
dps.usc.edu, emergency.usc.edu
Emergency assistance and avenue to report a crime. Latest updates regarding safety, including ways in which instruction will be continued if an officially declared emergency makes travel to campus infeasible.

USC Department of Public Safety - UPC: (213) 740-6000, HSC: (323) 442-120 – 24/7 on call
dps.usc.edu
Non-emergency assistance or information.