EE155L: Introduction to Computer Programming for Electrical Engineers

Course Units: 4

Syllabus (Version January 27, 2020)

Lectures: Mondays and Wednesdays 9:00-10:50am, in RTH 115
Labs: Fridays 2:00-4:50pm, in KAP 140
Webpage: http://blackboard.usc.edu

Students have the responsibility to stay current with material posted on website

Instructor: ***Sandeep Gupta***

Office: EEB 348
Tel: 213-740-2251
Email: sandeep@usc.edu – Start your email subject line as "EE-155: "
Office Hours:
Mon 2-3pm
Wed 2-3pm

Graduate Teaching Assistant: ***Bin Wang***

Office: PHE 320 (*only during office hours*)
Email: wang699@usc.edu – Start your email subject line as "EE-155: "
Office Hours:
Tue 5-6pm
Thu 2-3pm

Grader: ***Pruthvi Sumanth Kakani***

Email: pkakani@usc.edu – Start your email subject line as "EE-155: "
Office Hours: Via appointment (please send email)

Pre-requisite classes: ***None***

Registration requires EE major or approval of instructor.

Recommended Preparation: None. This is designed to be the first course in programming.

Catalogue Description: Algorithmic problem solving. Programming structure: datatypes, conditionals, loops, recursions, functions, arrays. File input/output and text manipulation. Algorithm complexity. Solving problems with C++, Python, and MATLAB.

Course Description: Electrical engineering students go on to a wide array of careers ranging from designers, system engineers, managers, executives, researchers, and even teachers, medical doctors, and lawyers. An essential trait for a successful career in any field is confidence and proficiency in using computers as problem solving tools. A fundamental part of becoming modern electrical engineer is learning how to program to solve engineering problems and gaining confidence and proficiency in learning new applications, programming languages, and computing platforms. In this class, students will gain experience in taking a given problem and developing a solution that can be realized via a computer program. Students will also gain experience working in small teams and reviewing the program designs of their colleagues.

Learning Objectives: Upon successful completion of this course a student will

1) Develop simple algorithms to solve given computational problems.

2) Develop good programming habits, including documentation, debugging, and testing.
3) Write computer programs using conditional, iterative, and recursive structures, as well as functional decomposition.
4) Select appropriate data structures (e.g., arrays and structures) and access methods (e.g., pointers).
5) Create programs that utilize file I/O.
6) Have a working knowledge of foundational C++ (C functionality in C++) to get ready for courses in Embedded Systems (EE109) and more advanced C++ programming and software engineering (EE355).
7) Have a basic working knowledge of MATLAB, including matrix and array computations.
8) Able to develop a basic understanding of Python, especially libraries relevant to EE (NumPy, SciPy).

Grading: Final course grades will be assigned by a combination of student score and some discretion of the instructor. We will have assignments, quizzes, and four exams (three midterms and one final). The weightage on these would be as follows.

1) 15% Assignments
2) 20% Quizzes (Surprise quizzes during lectures/labs; throughout the semester; best n - 1 out of n)
3) 10% Midterm-1 Exam (1 hour 45 minutes)
4) 20% Midterm-2 Exam (1 hour 45 minutes)
5) 25% Midterm-3 Exam (1 hour 45 minutes)
6) 10% Final Exam (1 hour or more)

Please submit all assignments on time. No make-up exams will be given, unless you have a major and documented emergency.

If you cannot take the final exam at the time and date designated by the University, you must notify the instructor at least one week before the last day to add/drop to see if the instructor can accommodate your schedule. In the case of a medical emergency, a signed letter from your doctor is required. This letter must include the telephone number of your doctor. In the case of emergency, the instructor must be notified via email as soon as possible.

Attendance: Lecture and lab attendance is strongly encouraged. Students are responsible for all material and information presented in lectures and labs. Also, surprise quizzes will be given during lectures and labs.

Exam Dates:

Midterm-1 Exam: Friday, February 7, 2pm (the first part of the discussion session)
Midterm-2 Exam: Friday, March 6, 2pm (the first part of the discussion session)
Midterm-3 Exam: Friday, April 10, 2pm (the first part of the discussion session)
Final Exam: Friday, May 8, 8am to 10am (https://classes.usc.edu/term-20201/finals/)

Course Activities: As is typically the case in industry positions, some (non-quiz and non-exam) activities will be collaborative. Some homeworks and programming assignments will be completed together in teams of two. Teammates may be assigned by the instructor and teams may be restructured several times throughout the semester. Early in the semester, students with previous programming experience are expected to mentor students for whom this is the first exposure to computer programming. Teammates may be asked to provide peer assessments of each other. Teams will present solutions and alternative designs during labs (similar to code review meetings in industry). Open-ended activities and self-discovery are valued and will be encouraged. Substantial support resources in the form of TAs will be provided.

Statement for Students with Disabilities: Any student requesting academic accommodations based on a disability is required to register with Disability Services and Programs (DSP) each semester. A letter of verification for approved accommodations can be obtained from DSP. Please be sure the letter is delivered to me (or to TA) as early in the semester as possible. DSP is located in GFS 120 and their phone number is (213) 740-0776. Additional information regarding DSP is available at https://dsp.usc.edu/.

Statement on Academic Integrity: USC seeks to maintain an optimal learning environment. General principles of academic honesty include the concept of respect for the intellectual property of others, the expectation that individual work will be submitted unless otherwise allowed by an instructor, and the obligations both to protect one's own academic work from misuse by others as well as to avoid using another's work as one's own. All students are expected to understand and abide by these principles. Scampus, the Student Guidebook, contains the Student Conduct Code in Section 11.00, while the recommended sanctions are described in Appendix A of the latest version of SCampus (https://policy.usc.edu/student/scampus/). Students will be referred to the Office of Student Judicial Affairs and Community Standards for further review, should there be any suspicion of academic dishonesty. The Review process can be found at https://sjacs.usc.edu/.

Emergency Preparedness/Course Continuity in a Crisis: In case of a declared emergency if travel to campus is not feasible, USC executive leadership will announce an electronic way for instructors to teach students in their residence halls or homes using a combination of Blackboard, teleconferencing, and other technologies.

Textbooks and References: There is a lot of information on programming on the internet that students are encouraged to utilize during this course and in their future careers. We will have one required textbook and rely on online materials for much of the course.

Required Textbook:

Cay S. Horstmann. Brief C++: Late Objects, Enhanced eText (ISBN: 9781119400424).
Additional references will be provided as the semester progresses.

<u>Weekly Course Schedule (reading from Horstmann shown in parenthesis):</u>

Week 1: C++ Program Basics (Ch. 2)
- Variable types and declaration
- Assignments and arithmetic
- Some library functions and header files
- Basic I/O
- Algorithmic thinking and flowcharts

Weeks 2 and 3: C++ Conditional and Loop Control (Ch. 3-4)
- if-else, switch statements in C++
- for, while, do loops in C++

Weeks 4 and 5: C++ Functions (Ch. 5)
- Why functions?
- Pass by reference vs. pass by value
- Recursion
- Variable scope

Week 6 and 7: C++ Arrays (Ch. 6)
- Array declaration, access, and passing to functions
- Multidimensional arrays

Matrix multiplication

Week 8: C++ Pointers (Ch. 7)
- Memory organization in computers
- Pointer concept
- Swapping arrays vs. copying arrays
- Strings and arrays as pointers
- Dynamic memory allocation
- Example: Sorting algorithms

Week 9: C++ Pointers (Ch. 7)
- Memory organization in computers
- Pointer concept
- Swapping arrays vs. copying arrays
- Strings and arrays as pointers
- Dynamic memory allocation
- Example: Sorting algorithms

Weeks 10 and 11: Computational problem solving (Lecture notes and class discussion)
- Manual problem solving
- From manual solutions to systematic approaches

Week 12: C++ Streams (Ch. 8)
- File management
- Reading, writing, appending text files
- Command line inputs and arguments
- Other file types

Week 13: MatLab (online resources and handouts)
- Syntax for assignment, conditionals, loops
- Working with arrays and matrices
- Applying function to arrays
- Recasting some C++ examples in MatLab

Weeks 14: How programs run and why we need so many languages ()
- Basic computer organization
- Languages in EE and the need for life-long learning of programming

Week 15: Review and applications
- How to teach yourself another language, say Python?
- What’s next – remainder of your UG program and beyond
- Review for final exam