ECEC-201 Advanced Programming for Engineers
Course Syllabus and Policies

Course Summary
This course will cover advanced usage and understanding of programming concepts using the C programming language. Primary focus is placed on the introduction of the low-level language features as they pertain to Computer Engineering majors, which will serve as a foundation for future embedded firmware and system level software authorship as well as a means to better understand the underlying mechanisms of program execution and memory organization used by modern computing systems.

Prerequisites: ENGR-131, ENGR-132, ECE-203, or CS-171

Mandatory Textbooks

C Programming: A Modern Approach, 2nd Edition
by K. N. King
W. W. Norton & Company
ISBN-13: 978-0393979503

Coverage
C coverage focuses on a low level understanding of the language as pertinent to Computer Engineering. Coverage of the C language is specifically constrained to the C89 language specification unless otherwise explicitly indicated.

- **Introduction to C** *(Week 1)* – **Virtual Address Space Basics:** the `.text` segment, the stack, shared libraries; **Function Mechanics:** parameter passing via the stack; return addresses; the stack trace; function return values via CPU register `eax`; the program counter CPU register; **C's Build Trajectory:** pre-processor intro; compiling to assembly; assembling to object files; linking object files and shared libraries to form executables; basic `gcc` and `GNU Make` usage; **Overview of Program Loading and Execution**. *Reading from King: Chapters 2 and 3*

- **Control Flow in C** *(Week 2)* – Logical expressions; **Selection Statements:** the `if-elseif-else` control flow construct, the `switch-case` control flow construct; **Loops:** the `while` construct, the `do-while` construct, the `for` construct; **Control Flow Modifiers:** the `break` keyword, the `continue` keyword, the `goto` keyword; **Functions:** function definitions, function declarations.
  Reading from King: Chapters 5, 6, and 9

- **Memory, Types, and Pointers** *(Week 3)* – The `char` type; the `int` types; `limits.h`; the `float` and `double` types; arrays; pointers; memory organization of arrays.
  Reading from King: Chapters 7, 8, 11, and 12

- **Strings, Structures, and Function Pointers** *(Week 4)* – Sequences of `char` elements; `null` termination (`\0`); introduction to `string.h`; introduction to `structs`;

the `->` operator; adding function pointers to a `struct` to hacking together "objects", understanding of how the implicit passing of the instance (i.e. `this` and `self`) came to be; simple API example
Reading from King: Chapters 13, 16, and 17.7

- **Dynamic Memory Intro & Singly Linked Lists** *(Week 5)* – The heap; working with dynamic memory via `malloc()` and `free()`; indexing dynamic memory block using array notation; implementing simple singly linked lists
Reading from King: 17.1–17.5

- **The `errno` System & Working with Files** *(Week 6)* – **errno**: Overview of the static memory location `errno`, the C standard library provided header `errno.h`, and using `errno` to retrieve error information; **Files**: The OS/userspace relationship when working with files, `FILE*` pointers, file modes, file I/O, and seeking within a file
Reading from King: Chapter 22 & 24.2

- **Linked Lists II & Structure Embedding** *(Week 7)* – Embedding a structure in another structure (i.e. nesting), accessing nested members from the outer structure, the `container_of()` macro; alternative singly linked list implementation using `container_of`; doubly linked lists, implementation using `container_of`
Reading: Handout

- **Hash Tables** *(Week 8)* – Using linked lists to build a simple hash table (i.e. dictionary or associative array), performance analysis, choosing a "good" hashing function
Reading from King: Handout

- **Embedded Style Fundamentals** *(Week 9)* – Introduction to the bitwise operators; bit masks; clearing, setting, and toggling individual bits; using `#define`s to make bit manipulation code easier to read and maintain; dealing with registers and memory mapped I/O
Reading from King: Chapter 20

Grading Policy
The grading in this course is based on lab assignments, homework assignments, and special projects. The cumulative grade is based on the following:

- 30% In-lab Programming Assignments
- 30% Take-Home Programming Assignments
- 40% Programming Projects

Grade Scale
Your course letter grade will be assigned as follows (may be curved):
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+: 65–69, D: 60–64, F: 00–59

Homework Assignment Policy
Homework is always to be submitted on or before the date for which it is due. Homework submitted after the due date will not be accepted and will be graded at 0 points. Homework must be submitted using Black Board Learn unless otherwise instructed. Assignments must never be submitted via e-mail.

Absentee Policy
Absence from examinations will be excused only under extraordinary circumstances such as medical or family emergencies. A missed examination without prior approval and without legitimate reason will be graded at zero points. An absence will be excused only if the student is able to provide legitimate documentation (such as a physician note). An absence from an examination with prior approval will require the student to take an alternate exam at a later time. Special examinations will not be held earlier or on later dates to accommodate, for example, flight schedules for overseas vacations.

Special Accommodations Policy
Students with disabilities requesting special accommodations must inform the professor within the first two week of the course. Such students requesting accommodations and services at Drexel University must present a current accommodation verification letter (AVL) to the instructor at least five days prior to receiving testing accommodations. AVL's are issued by the Office of Disability Services (ODS). For additional information, contact ODS at www.drexel.edu/ods, 3201 Arch St., Street, Suite 210, Philadelphia, PA 19104, 215.895.1401 (V), or 215.895.2299 (TTY).

Course Withdrawal Policy
Drexel's course withdrawl policy can be found at:
http://www.drexel.edu/provost/policies/course-withdrawal/

Course Change Policy
The instructor reserves to right to change the course during the quarter at his or her discretion. The changes, if applicable, will be communicated to the students verbally in class and reflected in the syllabus within the week of the change. Every effort will be made to not to change any course policy past the course withdrawal period and to collect student feedback prior to implementing such course change.

Academic Integrity Policy
Each student is expected to complete all assignments independently unless otherwise explicitly instructed. It is unacceptable to copy another student's work or solutions from any other source. Violators of this policy will be reported to the Office of Student Conduct and Community Standards (SCCS). Academic integrity violations could result in failure for the course or the assignment among other sanctions determined by the instructor. A second violation of the academic integrity policy will likely result in suspension.

Please refer to Drexel's Academic Integrity Policy for more information:
http://www.drexel.edu/provost/policies/academic-integrity/

Instructor

Dr. James A. Shackleford
shack@drexel.edu
Bossone 211
Office Hours: *Please e-mail and we will arrange a time*