CS24: Problem Solving with Computers II
Summer 2014 (Session C)

Instructor: Kyle Dewey (kyledewey@cs.ucsb.edu)
Teaching Assistants:
- David Adams (adams@mat.ucsb.edu)
- Soundharya Balasubramanian (soundharya@umail.ucsb.edu)

Website: http://cs.ucsb.edu/~kyledewey/cs24
Lecture: Tuesday / Thursday 9:30 AM - 10:45 AM in Phelps 3526
Lab: Wednesday 9:30 AM - 10:45 AM; 11:00 AM - 12:15 PM in Phelps 3525

Textbooks:
- Engineering Problem Solving with C, 4th Edition (by Delores M. Etter)
- C++ Plus Data Structures, 5th Edition (by Nell Dale)

Course Office Hours (all in Phelps 3518):
9 AM Monday and 11 AM Tuesday. Also available by appointment.

Course Description:
(In my own words) Building off of CS16, this course introduces fundamental data structures, along with operations on these structures. This course also introduces objects as a means to organize code.

(From the course catalog) Intermediate building blocks for solving problems using computers. Topics include data structures, object-oriented design and development, algorithms for manipulating these data structures and their runtime analyses. Data structures introduced include stacks, queues, lists, trees, and sets.

Prerequisites:
- CS16, with a grade of 'C' or better
- Math 3B, with a grade of 'C' or better (may be taken concurrently) OR AP 68 (Calculus BC) with a score of 3 or higher

Grading:
- 5% - Lab Attendance, 10 sessions @ 0.5% per session
- 17.5% - Exam #1, closed book, one hand-written page (double sided)
- 17.5% - Exam #2, closed book, one hand-written page (double sided)
- 30% - Final Exam - closed book, two hand-written pages (each double sided)
- 30% - Lab Assignments, 9 assignments @ 3.3% apiece

If your exam average is below 60% and is well below the class average, you will receive an 'F' in the class. This is to protect against students who lean on their partners too much, and whose tests demonstrate that they have not learned the material.

Course Outline: (subject to change)

Week	Day	Topic	Reading / Handouts
1	6/24	Review: Arrays, multiple files, `fgets`, command-line arguments	Etter Chap 5
1	6/26	Review: Pointers and allocation	Etter Chap 6
2	7/1	`struct`s, allocation, multiple files	Etter Chap 7
2	7/3	C++, objects, ADTs	Etter Chap 8 Dale Chap 2.1 - 2.4
3	7/8	Objects, constructors	Etter Chap 8 Dale Chap 2
3	7/10	Linked lists	Dale Chap 3.1 - 3.2
4	7/15	Linked lists	Dale Chap 3
4	7/17	Stacks, complexity	Dale Chap 2.6
5	7/22	Queues, review	Dale Chap 5.3 - 5.4
5	7/24	Exam 1	Etter Chap 7 - 8 Dale Chap 2 - 4
6	7/29	Complexity Analysis	Dale Chap 2.6
6	7/31	Recursion	Dale Chap 7.1 - 7.4
7	8/5	Recursion	Dale Chap 7.5 - 7.7, 7.10
7	8/7	Binary trees	Dale Chap 8.4 - 8.5
8	8/12	Binary trees	Dale Chap 8.6 - 8.8
8	8/14	Binary tree removes, heaps, priority queues	Dale Chap 8.6 - 8.7 Dale Chap 9.1 - 9.2
9	8/19	Exam 2	
9	8/21	Array-based implementations, hash tables	Dale Chap 8.8, 5.3, 10.3
10	8/26	Hash tables	Dale Chap 10.3
10	8/28	Exam 3	Dale Chap 1 - 9

On Pair Programming:
The labs will encourage students to do pair programming. In pair programming, one person is the "driver" who actually writes code, and another person "navigates" the driver through communication. People can switch between the two roles dynamically if they so choose. In my own experience, pair programming is excellent for tackling difficult problems that are simply too complex to keep contained in only one head. One person can focus on the details (usually the driver) while another can focus on the big picture (usually the navigator), and the end result is faster, less stressful development.

This, of course, is the ideal. This tends not to work well when two people of very different skill levels attempt it, and one of them is more anxious to "just get it done". In light of this, when selecting a partner for pair programming, it is best to select someone of approximately the same skill level.

On Collaboration:
In actual software development, very little work is done as an individual effort. Most modern software systems are simply too big to be undertaken in this style, so collaboration is key. This is one of the reasons why we are encouraging pair programming for some of the labs - this is a real, marketable skill.

That said, collaboration is **not** simply taking the credit for someone else's work. When this happens, **everyone** loses. The obvious loser is whoever did all the work. The less immediate loser is the one who took half of the credit for the work, because that person did not learn whatever the assignment was trying to teach. This can (and I have seen it happen during my own undergrad!) snowball long-term. Eventually the student who took the credit without working is truly completely lost, often without realizing it, and is held up entirely by the work of other students. This is a worst-case scenario, but in my experience it is far too common.

The point is this: collaboration means to work together, not to blindly take someone else's work or to give your own work away.

Attendance policy:
Attendance will not be taken in lecture, and there will be no graded assignments given during lecture. However, any material covered in lecture is ultimately **your** responsibility, regardless of whether or not the lecture was attended. Attendance will be taken for lab, and each lab counts towards 0.5% of your grade.

Due Dates / Late Policy:
For items turned in late, each person has 24 hours worth of "grace" time in total. For example, if someone were to submit the first lab 4 hours late and the second lab 6 hours late, then a total of 10 "grace" hours have been used. Both submissions would be accepted without incident, and there would be 14 "grace" hours remaining. For a pair submission, the grace time counts against both students individually. For example, if a pair submission were turned in 5 hours late, and one student had 20 hours of grace time

remaining and the other 5 hours, then both students receive credit. The first student ends up with 15 grace hours remaining and the second is left without any grace time remaining. Except in extenuating circumstances, submissions for students who have gone beyond their grace time will **not** be accepted.

A little background on this policy - the grace time is intended to be used as a sort of last-minute "oops" relating to a poor time estimate of (what should be) final touches. This policy tries to reduce the number of submissions hastily done just to meet a deadline, and to prevent issues of submissions that missed the deadline by a relatively small amount of time. It is not intended to be used as a way to extend the deadline for one assignment for a day, although it certainly can be used that way. Be forewarned: once it's gone it's gone, so use it wisely!

Extenuating Circumstances:
"Extenuating circumstances", for the purpose of this class, is defined as anything beyond our immediate control. In these cases, at my discretion I can grant an extension. To be absolutely clear, there is **no** guarantee that I will do so, and I am not obligated to grant them. For the things we can predict (e.g., trips), I expect to be contacted at least a week in advance. For the things we cannot predict (e.g., illness), I need official documentation explaining the situation (e.g., a doctor's note).

Communication Policy:
I have two office hours per week, though I may increase this to 3-4 if questions abound. I'm also available by appointment.

With email, assume that I will take at least 24 hours to respond. Typically my response time is much, much faster than this, but I do occasionally take this long. Historically, this has only been an issue the last hours before a project deadline, and only for students who started far too late. The point being: start early!

Academic Honesty:
In as few words as possible, cheating and plagiarism will **not** be tolerated. I understand that the temptation may be high ("it's just this one assignment" or "I just need this class"), but this is **no** excuse. At the very least, this is unfair to all the students who did not resort to such unethical means, who instead took the time and struggled through. I will be following UCSB's Academic Conduct policy on this (from http://www.sa.ucsb.edu/Regulations/student_conduct.aspx, under "General Standards of Conduct"), quoted below for convenience:

> It is expected that students attending the University of California understand and subscribe to the ideal of academic integrity, and are willing to bear individual responsibility for their work. Any work (written or otherwise) submitted to fulfill an academic requirement must represent a student's original work. Any act of academic dishonesty, such as cheating or plagiarism, will subject a person to University disciplinary action. Cheating includes, but is not limited to, looking at another student's examination, referring to unauthorized notes during an exam, providing answers, having another person take an exam for you, etc. Representing the words, ideas, or concepts of another person without appropriate attribution is plagiarism. Whenever another person's written work is utilized, whether it be a single phrase or longer, quotation marks must be

> used and sources cited. Paraphrasing another's work, i.e., borrowing the ideas or concepts and putting them into one's "own" words, must also be acknowledged. Although a person's state of mind and intention will be considered in determining the University response to an act of academic dishonesty, this in no way lessens the responsibility of the student.

Note that collaboration with a non-lab partner also constitutes cheating. Any incident of cheating **will** be reported. While this may sound steep, real-life cases of cheating (i.e. taking someone else's code without permission) have led to job termination and lawsuits among other things, so this is not unrealistic.