CS16: Problem Solving With Computers I
Syllabus, Fall 2009

Dept. of Computer Science, UC Santa Barbara

Printer friendly PDF version of this syllabus

Instructor: Dr. Phill Conrad, Lecturer (PSOE), Dept. of Computer Science (Joint Appointment, College of Creative Studies).

Website: http://www.cs.ucsb.edu/~pconrad/cs16

Lecture: 1-1:50pm MWF, Chem 1171.

Discussion sections: All held Friday in ESB 1003 (Cooper Lab)

Time	Main TA
10am-10:50am	
11am-11:50am	
noon-12:50pm	

Textbook: Engineering Problem Solving with C, 3rd Edition (by Delores M. Etter)

Contact Email: pconrad@cs.ucsb.edu

Course Office Hours are shown in the table below.

These office hours are in effect every day of the Fall Quarter at UCSB that classes meet (i.e. between 9/24 and 12/04, except University Holidays), unless otherwise announced on the course email list (and the Gauchospace main course forum.)

	Mon	Tue	Wed	Thu	Fri
Who	Phill Conrad	TBA	Phill Conrad	TBA	TBA
Where	HFH 1113	Phelps 1413	HFH 1113	Phelps 1413	Phelps 1413
When	2:15-4:15	TBA	4:15-5:15	TBA	TBA

Prof. Conrad is also available by appointment—email him to request an appointment. Include CS16 in your subject line, and indicate your available times.

TAs:

- Harry Presman <hpresman@umail.ucsb.edu>
- Shatra Sadhu <ssadhu@umail.ucsb.edu>
- Gorkem Yakin <gyakin@umail.ucsb.edu>

What this course is about

CS16 is an **intermediate course in programming**, focusing on **problem solving**, and fundamentals of software development.

What this course provides is a solid grouding in problem solving, and basic concepts of programming in C, so that you are well prepared to take CS24 and CS40 (which are the next CS courses in the curriculum.)

I say that the course "provides an opportunity", because you will get a solid grouding in problem solving and C programming **only** if **you** put a lot of time and effort into this course—that is true no matter how much thought and attention I put in my lectures, assignments, and exams.

Are you ready for CS16?

CS16 should not be your first programming course!

This course assumes you have already done *some* programming.

No prior knowledge of any **particular** programming language is required. We will use the C programming language in this course to teach problem solving—however, we do **not** require any prior knowledge of C.

However, we do assume you are already a beginning programmer in **some** language such as Python, Java, C, C++, C#, Visual Basic, JavaScript or ActionScript. You should have already taken a course in programming prior to enrolling in CS16—either in high school, and a previous college or university, or perhaps ENGR3, CMPSC5 or CMPSC8 here at UCSB.

You should already be comfortble with concepts such as:

- variables and assignment statements (e.g. x = x + 1;)
- function definitions and function calls (e.g. `celsiusTemp = convertToCelsius(fahrenheitTemp);`)
- if/else for making decisions, and expressions involving and, or and not
- loops, such as for loops and while loops
- arrays (or a similar structure such as lists in Python)

If you are not comfortable with all those things, you should drop CS16 from your schedule and try to enroll in CS8 instead.

Note: prior experience with web pages and HTML is **NOT** enough background to be successful in this course.

If you don't have the proper background for CS16, please don't enroll just because:

- it "fits your schedule better" than CS8 (bad idea!)
- there was room in CS16 on GOLD, but CS8 was full (bad idea!)

If you really should be in CS8 but are having scheduling issues, please see Benji Dunson, the undergraduate advisor for CS, who is available during normal business hours in the CS office, on the 2nd floor of Harold Frank Hall. He can help with authorization to enroll ("add codes"), and other similar problems.

Learning to program: the swimming/guitar/painting analogy

You cannot learn to swim, play guitar, or paint from a textbook or a lecture. You can only

- learn swimming by spending many hours in the pool,
- guitar by spending many hours playing the instrument
- painting, by spending many hours putting brush to canvas.

The same is true of the type of problem solving involved in programming. Programming is not a serious of facts to be memorized—you cannot "cram" for a computer science exam. You must practice, practice, practice.

This course may have a heavy workload

As a result, the workload in this course may feel heavy. It may even feel unreasonable compared to your other courses. However, I assure you that it is not unreasonable, given the goal of making you an solid problem solver, and preparing you for CS24.

What you need to learn

So, what is it that you need to know to be ready for CS24?

At the very least, mastery of basic computer organization and programming constructs:

- computer organization: memory, CPU, I/O devices
- data representation and number systems: binary, octal, hex, binary arithmetic
- variables and expressions, statements
- control structures: conditionals and iteration (i.e. loops such as for loops and while loops)
- functions (parameters, recursion)
- primitive and composite data types, e.g. arrays and structs
- basic operating system and debugging tools.

Finally, here is a list of some things you need to begin to appreciate that are more subtle, but perhaps even more important in the long run:

- problem solving skills
- software development "team" and "communication" skills

Here's a bit more detail about these aspects of the course:

Problem Solving

Problem solving is something that is difficult to "pin down". We can't give you a step-by-step process for doing it—if we could, it wouldn't be problem solving! It is that part of developing software that involves seeing how to combine all the pieces you've learned about in a new way in order to make a piece of software work.

It is something that can't be learned from a lecture or a book—though it can be learned! The only way for you to learn it is for us—your instructors and TAs—to offer you many opportunities to try doing it. You'll make mistakes, and then have flashes of insight, and learn from those. Eventually, both your skill and self-confidence will increase.

Team and Communication Skills

In terms of "the real world"----i.e. what employers want us here at UCSB to "do better at"---these are the things they almost always bring up. Companies that hire programmers don't complain that UCSB grads lack technical skills, but they do wish they were better at these things:

- Understanding the importance of making your code easy for other people to read, understand and modify
- Being able to talk about your code, and write about your code.
- Understanding that code lives for a long time—you don't just write it, and then walk away from it.
- Understanding the value of working with others on software, rather than all by yourself (more on this below).

These last four points are often difficult to convey in a 10 week class at the introductory/intermediate level, but we are going to do our best to try!

A special note about collaboration

As mentioned above, one of the things we really want to convey in this course is that real-world software development is very seldom an 'individual sport'—is it much more often a 'team sport'. Companies want to hire CS and CE graduates that know how to collaborate with others on producing software.

In the CS Department at UCSB, we understand the value of this. However, it puts us in a tricky position.

On the one hand, we want to encourage working together in ways that help you develop your skills and teamwork, and help you understand that programming can be a social, collaborative, creative activity—not something done only by loner nerds in cubicles. The sooner you start with activities such as pair programming, code reviews, and other collaborative software development activities, the more skill you'll develop, and the sooner you'll be ready for the real world. Plus, for many people, working together with others is a lot more enjoyable and fun than being a loner.

On the other hand, we need to avoid any situations where freeloaders are "coasting" through courses by leaning too much on others—never developing independent skills as programmers. This situation creates huge problems. Mostly it is damaging to the freeloaders themselves, who eventually crash and burn, perhaps far too late to choose another career path without significant difficulty. However, it also creates problems for everyone else—some hardworking students become demoralized by the unfairness of it all, and the value of a UCSB education is diminished by the freeloaders' lack of accomplishment.

Thus, we must strike a balance.

Our emphasis on collaboration means:

- We will try to create opportunities for you to work in pairs on assignments—in some cases, we may even require it.
- We will try to create opportunities for you to develop skills at talking about and reflecting on your own code and other people's code in small groups (code reviews).
- Some in-class assignments will permit discussion with other students.

It doesn't not mean, however:

- That you can "just copy" homework or code from others and claim it as your own work.
- That you can work together on assignments where you've been specifically told not to.

The bottom line:

- We'll try to be very specific about what kinds of collaboration are permitted, and what kinds of collaboration are not permitted, and are considered academic dishonesty.
- If you are not sure about whether some kind of collaboration is permitted or not, it is your responsibility to **ask questions.**

A final note: the emphasis on collaboration in this course does not necessarily extend to other CS courses you make take in the future.

- Each course will have its own policies, and the default policy is still: **no collaboration.**
- Please be sure you understand each instructors policy on collaboration carefully, and don't assume it will be the same as that from previous courses.

Catalog Description

Fundamental building blocks for solving problems using computers. Topics include basic computer organization and programming constructs: memory, CPU, binary arithmetic, variables, expressions, statements, conditionals, iteration, functions, parameters, recursion, primitive and composite data types, and basic operating system and debugging tools.

Official Prerequisites/Restrictions (from Catalog)

- *Prerequisite: Math 3A.*
- *Students with no experience with computer programming are encouraged to take Computer Science 5 or 8 before Computer Science 16.*
- *Not open for credit to students who have completed Computer Science 10.*

Grading

To determine your final letter grade, I will calculate two numbers:

Weighted course average	50% Assignments/Quizzes/Homework + 30% Midterm Exams (2 at 15% each) + 20% Final Examination
Weighted exam average	30% Midterm Exams (2 at 15% each) + 20% Final Examination

I will then use the following table. This is a conventional 10 point scale with +, - with the lower three and upper three points of each range representing plus/minus. It also enforces a policy that your final course grade can be no more than one letter grade higher than your exam average.

to earn this letter grade	your weighted course average must be at least	AND, your weighted exam average must be at least
A	93	83
A-	90	80
B+	87	77
B	83	73
B-	80	70
C+	77	67
C	73	63
C-	70	60
D+	67	57
D	63	53
D-	60	50
F	weighted course average below 60 **OR** weighted exam average below 50	

Curving: The grade scale above represents the *minimum* letter grade you will be assigned—at the instructor's discretion, the grading scale may be altered *in the students' favor* if this will be better reflect the students' mastery of the material. Thus, *if* there is a "curve", it will be applied at the *end*, not to individual assignments.

A+ grades: These may be awarded to the very best performing students in the class—but the cutoff for A+ grades will be determined at the end of the course at the discretion of the instructor (there is no pre-determined cutoff---an average of 97 or more doesn't guarantee you an A+ grade.)

Policies

Attendance

This course moves quickly. So attendance is very important.

We'll be trying to master the material from about 14 chapters in the book, at about 2 chapters per week. We need to go at that pace, because we'll lose a couple of weeks to exams, and the last few lectures the quarter, you can't really start anything new, because there isn't time to put it into practice with programming assignments. If you don't put it into practice, you aren't very likely to learn it in any way that is going to stick with you, so there isn't much point in just "going through the motions".

As a result, **there will be something you have to turn in at almost every class**. In this way, attendance is taken, and required.

These things you have to turn in will be a combination of in-class activities, and homework completed outside of class, but handed in on paper during class.

Quizzes may occur at anytime, announced or unannounced. Missed quizzes may not be made up, except per the "personal day/sick day" below—if you miss a quiz for any reason, and have already used your personal day/sick day, you will have to make up the points with extra credit.

Thus **attendance is required**, and **reading the assigned readings is required.**

Missing in-class activites.

If you miss a class, you miss the opportunity for the points on that in-class assignment, or homework that was due. Period.

There is no makeup, except for

- excused absences arranged and agreed to by the instructor **in advance**, for official UCSB activities
- one "sick-day/personal day" per student, per quarter (see below)

Sick day/personal day

To make up an assignment from a "sick-day/personal-day", you must email me within 48 hours of the absence, to make an appointment to make up the assignment during the next scheduled office hours following your absence (or at an appointment time to be negotiated, if you have a conflict with those hours.) This make up must happen within one week of the absence, or 24 hours before the final exam, which ever is earlier.

In rare cases, if there is a documented family emergency, documented extended illness, documented required court appearance, or other situation beyond the students' control (with documentation) the instructor may grant additional make up days entirely at the instructor's discretion—but this is **not** a guarantee or a right.

Academic Honesty

You should read and understand the UCSB policy on academic honesty listed below. You should also understand that I take academic honesty and personal integrity very seriously, and will do my best to uphold and enforce this UCSB policy. (Also see the section on collaboration, earlier in this syllabus.)

> 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. Using or attempting to use materials, information, study aids, or commercial "research" services not authorized by the instructor of the course constitutes cheating. 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.
>
> *(Section A.2 from: **http://www.sa.ucsb.edu/regulations**, Student Conduct, General Standards of Conduct)*

Accomodations for disabilities

Information about how UCSB supports students with disabilities is available at the campus ADA website: http://www.ada.ucsb.edu. If you require any special accomodations due to disabilities, please let me know as soon as possible. You may contact me by email to request an appointment: .

Disclaimer

This syllabus is as accurate as possible, but is subject to change as the instructors discretion, within the bounds of UC policy.