UNIVERSITY of CALIFORNIA at Berkeley
Department of Electrical Engineering and Computer Sciences
Computer Sciences Division

CS61A
Course Info

Kurt Meinz
Summer 2005

CS61A: The Structure and Interpretation of Computer Programs
General Course Information

1 Introduction

The CS 61 series is an introduction to computer science, with particular emphasis on software and on machines from a programmer's point of view. This first course concentrates mostly on the idea of *abstraction*, allowing the programmer to think in terms appropriate to the problem rather than in low-level operations dictated by the computer hardware. The next course, CS 61B, will deal with the more advanced engineering aspects of software—on constructing and analyzing large programs and on techniques for handling computationally expensive programs. Finally, CS 61C concentrates on machines and how they carry out the programs you write.

In CS 61A, we are interested in teaching you about programming *per se* rather than any programming language in particular. We consider a series of techniques for controlling program complexity, such as functional programming, data abstraction, object-oriented programming, and deductive systems. Of course, to get past generalities you must have programming practice in some particular language, and, in this course, we will use Scheme, a dialect of Lisp. This language is particularly well-suited to the organizing ideas we want to teach. Our hope, however, is that once you have learned the essence of programming, you will find that picking up a new programming language is but a few days' work.

2 Do You Belong Here?

The summer session version of this course is a bit different from the regular semester version. We cover all of the usual material, but we do it in **half** the time. This makes the course *very fast*. If you fall behind, you will find it almost impossible to catch up. At the same time, the summer course has no restrictions on enrollment. Anyone, regardless of prior experience may enroll in the course (until it fills.) We encourage anyone who's curious or interested to take this course, even if they aren't computer science majors!

With that being said, this course will be difficult and time-consuming. Your nominal classroom hours are roughly 12hrs/week – however, you can expect to be spending, at the very least, another 20hrs/week on readings and assignments. If you have other time commitments, such as a summer job or another summer course, you may find yourself stretched too thin. In short, this course will be like a full-time job, so please plan accordingly.

This course expects some logical sophistication, but does not actually require any prior programming experience. During the regular semester, Math 1A is a corequisite for 61A, and there is generally a placement exam to test whether or not you are familiar with *recursion* or *induction*. (For examples, go to `http://www-inst.eecs.berkeley.edu/~cs61a/miscellaneous/entrance.html`.)

We have found that 80% to 90% of 61A students have had significant prior programming experience, and that students without such experience are at an initial disadvantage. There is no need for you to be familiar with any particular programming language, although if all of your experience has been in BASIC then you probably haven't used recursion. In addition, the computer labs for the course use UNIX machines. You may find it time-consuming and sometimes difficult to do the labs and homework if you have not spent time becoming familiar with UNIX.

Therefore, it is up to you to decide if you are prepared for this course. Check out the course materials yourself, and play around with the labs and homework. My advice is to take the risk and get out as much as you possibly can! If you are still unsure, you can speak to me about it, however, if you ask my opinion, I will probably say that you should take it because the course is wonderful and you will learn a great deal from taking it (regardless of your final grade).

If you don't feel ready for 61A, we recommend that you take CS 3, which is a Scheme-based introductory programming course, or CS 3S, the self-paced version. CS 3 and 3S are directed primarily at students who are not Computer Science majors, but they are also designed to serve as preparation for 61A. You could then take 61A next semester. If you are interested in learning how to program specifically in C or Java, there are engineering courses to teach you these courses, and they will server you better than this course.

If you are not strongly interested in computer *programming* at all, but instead want to learn how to *use* computers as a tool, you should consider IDS 110, a course that presents a variety of personal computer software along with a brief introduction to programming.

If you have substantial prior programming background, you may feel that you can skip 61A. In most cases, we don't recommend that. Although 61A is the first course in the CS sequence, it's quite different from most introductory courses. Unless you have used this same textbook elsewhere, I think I can promise that you won't be bored. If you're not convinced, spend some time looking over the book and then come discuss it with me. Instead, perhaps your prior experience will allow you to skip 61B or 61C, which are more comparable to courses taught elsewhere. See Mike Clancy in the CS department about this.

3 Course Materials

The textbook for this course is *Structure and Interpretation of Computer Programs* by Abelson, Sussman, and Sussman, second edition. It should be available in the textbook section of the ASUC bookstore and other local textbook sellers. **You must get the 1996 second edition! Don't buy a used copy of the first edition.** A paperback version containing all necessary chapters of version 2 may also be available used at the same books stores. If you cannot afford or don't want to buy the book, copies of it are on reserve at the Engineering Library. Also, the **entire** book is readable online. The URL is given later in this document and on the course website.

In addition to the textbook, there is a reader containing necessary materials, including most assignments, information on our computing facilities in general, and about the Scheme language. You can buy the reader at CopyCentral, 2483 Hearst Avenue (at Euclid.) The summer's reader is unlike the normal term's readers, so don't borrow your housemate's old copy. All of the most important material in the reader will also be available on the course website, so, if you really don't want to buy the reader, you don't have to. However, it has been our experience that most students prefer to purchase the reader.

We have also listed an optional text for the course. This book really is optional! Don't just buy it because you saw it on the shelf. The optional text is *Simply Scheme*, by Harvey and Wright. (Brian Harvey is a professor here.) This is usually used as the textbook for CS 3; it gives a slower and gentler introduction to the first five weeks of 61A, for people who feel swamped here.

If you have a home computer, you may want to get a Scheme interpreter for it. The Computer Science Division can provide you with free versions of Scheme for Linux, Windows, or MacOS. The distribution also includes the Scheme library programs that we use in this course. For more information on how to get your home computer to work well with the course materials, check the 'resources' section of the web site.

The course reader includes the notes for the entire semester, and I will make my presentation slides available either immediately before or immediately after lecture. These notes are provided so that you can devote your efforts during lecture to thinking, rather than to frantic scribbling. In addition, any materials used during lecture but not provided in the reader will be made available on-line.

4 Enrollment—Laboratory and Discussion Sections

Summer session is 8 weeks, with every week packing in two standard course weeks. This course is normally structured so that there is one discussion and one lab meeting each week; but we must pack in both into the first two days of the week, and again, both into the last two days of the week. Generally, the lab portion occurs some time after Monday's lecture and again sometime after Wednesday's lecture. The discussion sections meet after lecture on Tuesdays and Thursdays. You will also need to spend additional time working on the computers in the Soda Hall labs. For most weeks, labs will meet in our laboratory room, 271 Soda Hall and the discussions will be in 310 Soda. Occasionally there may be two lab sessions and no discussions. Be sure to check the website and pay attention in lecture.

The discussion and lab sections are run by our Teaching Assistant, Jeff. We anticipate some rearrangements during the first week in response to oversubscribed or undersubscribed sections. **If you are waitlisted**, you should communicate via email with Jeff about your situation. Please be in a definite discussion section by the end of this week, though, because some of the coursework will be done in groups of two students; you are not allowed to form a group with someone in a different section.

You must have a computer account on the 61A course facility. You must set up your account *before Noon on Wednesday, June 22* because that is how we know who is really in the class. Account forms will be distributed in the LAB SECTIONS. The first time you log in, you will be asked to type in your name and reg card number, if you have one. Please follow the instructions carefully. **Be sure to remember the "secret code" you use for registration – you will use that secret code to check your grades on-line.** You must get your account *and log into it* no later than **12:01 PM Wednesday** so that we have an accurate class count. Everyone MUST log in by Wednesday Noon (or have made special arrangements with their TA) **OR YOU WILL BE DROPPED** from the course and someone on the waitlist will take your place!

Some of you have personal computers and may want to do the course work at home. This is fine with us, although you'll have to be careful to install the class Scheme library on your home computer to make your computer's version of Scheme behave like the modified one we use in the lab. In any case, though, you must get a class account even if you intend never to use it.

If you get a class account and then decide to drop the course, please let me know *immediately* so that we can admit another student. Thank you.

5 How to get the most from this course

We recognize that everyone's style of learning is unique. Some students are excellent at studying–they work hard, and are extremely diligent. They do all the readings conscientiously, and work all the problems. Some students are incredibly quick, and get by doing little of the reading, even less of the homework, and still ace the tests. Some students learn best by listening to lecture, and discussing it with their friends and TAs. Some students are aiming for the A+, others just to get by with a passing grade. Usually, students are some of each of these types, or are sometimes one, sometimes another. Since everyone's style is their own, we try to have as many opportunities to learn this material as possible. Therefore, use them all, and learn what works best for you.

That said, we do enforce certain types of interaction. In this course, we encourage and REQUIRE that you learn to work together in groups for certain assignments. This means you will need to learn how to work with people whose strengths are not your own. (This is of course the best thing a group can provide!) It also means you will learn how to work with people whose style you find difficult. But overall, you will learn best by learning to collaborate, and helping each other when one is not getting the material.

Different people solve problems differently; there are often many right answers to the problems in this course. And of course, what you find easy, your friend may find hard, and vice versa. Therefore, the best way to learn is to talk with other people, and ask them questions when you are stuck. Even if you think you understand everything, you will learn the material better if you have to try to explain it to someone else. In addition, learning how to think about the problems in many different ways will solidify your understanding

of this material.

Finally, is it possible that some of you feel uncomfortable telling others when you don't understand something. Many of us find it hard to ask questions–all the more reason to overcome this fear early! The ability to ask for help is a wonderful strength that will serve you well in life. Throughout this course, I will try to encourage you to ask each other, the staff, and myself for help.

6 Information Resources

Jeff, the readers, and I available to answer questions. You may drop in during office hours, make appointments for other times, or communicate with us by email. Feel free to visit any of the staff. You may find that hearing different people's explanations helps if at first you do not understand some material.

For technical questions about the homework or projects, or administrative questions such as missing homework grades, send electronic mail to your reader. You can also send mail about intellectual questions to me, but if it's about grades I'll just refer you to your TA.

In addition, there is an electronic bulletin board system that you can use to communicate with other 61A students and staff. The ucb newsgroup can be read only from machines in the berkeley.edu domain, so if your net connection is though a commercial ISP then you must log into a lab machine to read the newsgroup or try this:

`http://www-inst.eecs.berkeley.edu/connecting.html`

Please do not send electronic mail to every student individually! That would waste a lot of disk space, even for a small message. Use the newsgroup instead. Electronic mail is for messages to individuals, not to groups.

There is a web-based reader for the cs newsgroup available from the course homepage, located at

`http://www-inst.eecs.berkeley.edu/~cs61a`

The web page for the textbook, with additional study resources, is

`http://www-mitpress.mit.edu/sicp/sicp.html`

There are also web pages for the Scheme programming language:

`http://swissnet.ai.mit.edu/scheme-home.html`

`http://www.schemers.org/`

Additional information to help you in studying, including hints from the course staff and copies of programs demonstrated in lectures, is available at the course website.

7 Computer Resources

The computing laboratory in 271 Soda Hall consists of about 35 SunRay terminals connected to a Sun Solaris server. This is our primary lab room, although the CS 61A accounts can also be used from any EECS Instructional lab in Soda or Cory Hall.

The lab in 271 Soda is normally available for use at all times, but **you need a card key for access to the lab**; to get a card key, stop by the 3rd floor office of Soda Hall and fill out a form for a card key. You will need a $20 deposit to get the card key. The card key will give you access to the 2nd and 3rd floors of Soda Hall so that you may enter at any time, day or night. Do this today! During scheduled lab sessions, only students enrolled in that particular section may be in the lab. Therefore, you might need to use the other Soda Hall labs to work on homework outside of class. In particular, 273 Soda Hall should be at your disposal at all times. When sections are not in session, any 61A student may use any of the 2nd floor labs on a drop-in basis. If there are no free workstations, please feel free to ask anyone who is not doing course work to leave. In particular, *game playing is not permitted*. We are relying on social pressure to discourage abuse (such as stealing the chairs or monopolizing a workstation for six hours during prime time to play chess). Therefore, do not feel embarrassed to apply such pressure.

These machines use the Unix operating system, a timesharing system that is quite different from the

microcomputer systems you have probably seen elsewhere. The course reader includes introductory documentation about Unix and about Emacs, the text editing program we are recommending for your use. (It is one of several Unix text editors; you'll find that everyone has his or her own favorite editor and hates all the others.) Although the use of Unix is not extensively taught in 61A lectures, it will be extremely worthwhile for you to spend some time getting to know how the system works.

If you have a home computer and a modem, you may wish to use your class account remotely. If so, you are encouraged to use a commercial Internet Service Provider to connect to the campus; several companies offer student rates. Again, check out

`http://www-inst.eecs.berkeley.edu/connecting.html`

In addition, you should know that, on occasion, our file servers go on the blink. You can detect this situation by noticing that your terminal has suddenly stopped typing characters or you get a message along the lines of "`NFS server not responding...`". If this happens to you (and it will at least once!), don't panic; usually the server is back within minutes or hours with your data intact. Please do not put yourself in a situation where a couple-hour server crash will prevent you from completing your project on-time. "How can I avoid such a horrible situation?" you may ask. By starting (and finishing) your assignments early, of course!

8 Reading, Homework and Programming Assignments

You should try to complete the *reading* assignment for each week **before** the lecture. You will have four subsequent class meetings (two lectures and two discussion/lab sections) to help you understand the readings. Ideally, you would work in lab and afterward on the exercises, and then complete them the next day after section. If you're efficient, you'll then have that night to read the next reading assignment.

Every week there will be problems assigned for you to work on, many of which will involve writing and debugging computer programs. These assignments come in three forms:

- **Laboratory exercises** are short, relatively simple exercises designed to introduce a new topic. Most weeks you'll do these during the scheduled lab meeting following Monday and Wednesday's lecture. Labs are worth a small but nonnegligible number of points, and are typically checked off by Jeff during the lab period.
- **Homework assignments** consist mostly of more difficult problems designed to solidify your understanding of the course material; you'll do these whenever you can schedule time, either in the lab or at home. You may be accustomed to homeworks with huge numbers of boring, repetitive exercises. You won't find that in here! Each assigned exercise teaches an important point.

  There are two homework assignments per week, but both are due on the Sunday after they are assigned. These assignments are included in the course reader and the course homepage. You are encouraged to *discuss* the homework with other students. Specific Homework requirements and grading policies are below.
- **Projects** are larger assignments intended both to teach you the skill of developing a large program and to assess your understanding of the course material. There are four projects during the term, and you'll work on some of them in groups. Specific project requirements and grading policies are listed below.

Everything you turn in for grading must show your name(s) and your computer account login(s)! Please cooperate about this; make sure they're visible on the *top* of the files you turn in, not buried somewhere in a comment or a function.

9 Testing and Grading

The grading policy of the course has these goals: it should provide a reasonably accurate measure of your understanding of the material; it should minimize competitiveness and grade pressure, so that you can focus

instead on the intellectual content of the course; and it should minimize the time I spend arguing with you about your grades. To meet these goals, your course grade is computed using a point system with a total of 300 points:

16 labs	@	1 point each	=	16 pts
15 homeworks	@	5 points each	=	75 pts
2 mini projects	@	7 points each	=	14 pts
2 larger projects	@	10 points each	=	20 pts
3 midterms	@	35 points each	=	105 pts
1 final			=	70 pts
--				---
39 assignments				300 pts

There will be three midterms (set for the end of the third, fifth, and seventh weeks of the term) and a final. The exams will be open book, open notes. (You may not use a computer during the exam.) In the past, some students have worried about time pressure, so we'll hold the midterms on Fridays 'round Noon (Room TBA) instead of during the lecture hour. My goal will be to write one-hour tests, but you'll have at least two hours to work on them. The relatively large number of midterms is meant to help you learn to take tests, and to reduce your anxiety about ruining your grade by having a bad day. In this course, the later topics depend on the early ones, so you must not forget things after each test is over!

Each letter grade corresponds to a range of point scores: 270 points and up is an A, 260–269 is A-, and so on by steps of ten points to 170–179 points for a D−.

		A	270-300	A-	260-269
B+	250-259	B	240-249	B-	230-239
C+	220-229	C	210-219	C-	200-209
D+	190-199	D	180-189	D-	170-179

This grading formula implies that **there is no curve**; your grade will depend only on how well you (and, to a small extent, your partner) do, and not on how well everyone else does. (If everyone does exceptionally badly on some exam, I may decide the exam was at fault rather than the students, in which case I'll adjust the grade cutoffs as I deem appropriate. But I won't adjust in the other direction; if everyone gets an A, that's great.)

Extra credit: Extra credit will be granted at the sole election of the staff and is typically reserved for persons who make a substantical and interesting (in the conceptual sense) addition to an assignment. I will mention opportunities for extra credit in class.

Exam regrading: If you believe we have misgraded an exam, there will be a regrading policy posted on the course website. At the very least, your entire exam will be regraded, so be sure that your score will really improve through this regrading! By University policy, final exams may *not* be regraded; to make up for this, we will grade every final exam twice. Final exams may be viewed at times and places to be announced.

Incomplete grades will be granted only for dire medical or personal emergencies that cause you to miss the final, and only if your work up to that point has been satisfactory.

10 Homework and Project Policies and Grading

In contrast to prior semesters, homework in this course will be done independently. You and your friends are encouraged to discuss the problems among yourselves, but the work that you turn in must be written and tested by you alone. Both of each week's homework assignments are due at 8:00 PM on the following Sunday. Both **homework sets must be submitted electronically** unless otherwise noted.

The purpose of the homework is for you to learn the course, not to prove that you already know it. Therefore, although the weekly homeworks will graded on correctness, you will be afforded an opportunity to recover points by improving your understanding of the material. **If you receive less than 90/100 credit on a particular homework, you can sign up for a face-to-face session with your reader.**

During this session, you will have an opportunity to convince your reader that your understanding of the material has improved. If you show sufficient improvement, the reader may adjust your score upwards. Sign-up sheets for the face-to-face sessions will be posted in the laboratory (and perhaps online). **Please bring a paper copy of your homework to the sessions!**

The four programming projects are graded on correctness and style. The first two projects are to be done individually, and the last two in groups of exactly two. The last two projects are larger, and your group will work on a single solution, but the problems within each project are divided into two sets, and each of you will work on one set.

The latter two projects will probably include face-to-face grading with your reader. The reader will ask questions of each member of your group, and you will be graded by ALL of the group's members' ability to answer correctly. Therefore, you must work together to ensure that your partner understands the entire project.

Your group will turn in *one copy* of each project, with both of your names and logins listed on it. **The programming projects must be turned in online as well as in the homework box;** the deadline is usually 11:59 PM on the second Tuesday after it is assigned (i.e. you have two weeks for each project), but there will be some exceptions. You'll get further instructions about this when the time comes.

Online turnin: You must create a directory (you'll learn how to do that in lab) with the official assignment name, which will be something like `hw3` or `proj1`. Put in that directory all the files that you want to turn in. Then, while still in that directory, give the shell command `submit hw5` (or whatever the assignment name is). We'll give more details in the lab.

Paper turnin: There are boxes with slots labelled by course in room 283 Soda Hall. (Don't put them in my mailbox or on my office door!) What you turn in should include transcripts showing that you have tested your solution as appropriate.

11 Collaborative Learning Policies and Cheating

We encourage collaboration. It is the best way to learn and keep up with the wealth of material you are expected to cover. At the same time, cheating is not permitted. Sometimes the line between collaboration and cheating doesn't seem so easy to articulate, so we've tried to come up with very clear and enforceable rules so that you know what is expected and aren't uncomfortable collaborating, and, at the same time, so that those who break the rules can be held accountable.

Unlinke the degree of collaboration allowed and expected on homeworks and labs, the tests in this course must be your own, individual work. I hope that you will work cooperatively with your friends *before* the test to help each other prepare by learning the ideas and skills in the course. But during the test you're on your own. The EECS Department Policy on Academic Dishonesty says, "Copying all or part of another person's work, or using reference materials not specifically allowed, are forms of cheating and will not be tolerated." (61A tests are open-book, so reference materials are okay.) The policy statement goes on to explain the penalties for cheating, which range from a zero grade for the test up to dismissal from the University, for a second offense.

For the programming projects, copying others' work, whether from your friend who took the course last semester or from other current students in other groups is cheating. If you don't know how to do something, it's better to leave it out than to copy someone else's work. If you do learn something from someone else, and understand it now, then cite it as theirs. But be prepared to back up that you understand it without them around. If you do not cite it, it is considered plagiarism, and is again, cheating.

It is highly unlikely that different people would arrive at the exact same solutions on their own. We do have programs to test for code similarity – and these programs are smart enough to know when only the variable names have been changed. Don't cheat–you do a disservice to yourself, to those you copy from, and ultimately, to the whole course as time is taken away from preparing lectures and answering questions to deal with cheaters.

For the homework assignments, before you develop your solutions to the problems you are encouraged to discuss it with other students, in groups as large or small as you like. **When you turn in solutions, you**

must give credit to any other student(s) who contributed to your work. This does not mean e.g. 16 of you should turn in precisely the same work. It means that you may talk about it, work it out, try it, and then each person writes it up on their own. Working on the homework in groups is both a good way to learn and a lot more fun! If you take the opportunity to discuss the homework with other students then you'll probably solve every problem correctly.

In my experience, most students who cheat do so because they fall behind gradually, and then panic at the last minute. Some students get into this situation because they are afraid of an unpleasant conversation with an instructor if they admit to not understanding something. I would much rather deal with your misunderstanding *early* than deal with its consequences later. Even if the problem is that you spent the weekend stoned out of your skull instead of doing your homework, please overcome your feelings of guilt and ask for help as soon as you need it.

If you are still unclear on the cheating policy, ask yourself this: in all of your talking with other students, did you UNDERSTAND the solution, or did you merely write down what someone else told you? If you didn't understand, that you aren't doing the work yourself– not honestly. Again, it is better to have the answer wrong, or only partially right than to rely on someone else's answer. (Often because they too could be wrong!)

Working cooperatively in groups is a change from the traditional approach in schools, in which students work either in isolation or in competition. But cooperative learning has become increasingly popular as educational research has demonstrated its effectiveness. One advantage of cooperative learning is that it allows us to give intense assignments, from which you'll learn a great deal, while limiting the workload for each individual student. Another advantage, of course, is that it helps you to understand new ideas when you discuss them with other people. Even if you are the "smartest" person in your group, you'll find that you learn a lot by discussing the course with other students. For example, in the past some of our best students have commented that they didn't *really* understand the course until they worked as lab assistants and had to explain the ideas to later students.

If some medical or personal emergency takes you away from the course for an extended period, or if you decide to drop the course for any reason, please don't just disappear silently! You should inform the other members of your group, and your TA, so that nobody is depending on you to do something you can't finish.

Penalties for cheating: Generally, the penalty for cheating on any assignment will be, at the very least, a zero on the assignment and will result in a notice being sent to the Office of Student Conduct. Further offenses and particularly egregious forms of cheating (like selling answers) will be dealt with more severely.

12 Lateness

A programming project that is not ready by the deadline may be turned in until 24 hours after the due date. These late projects will count for 2/3 of the earned score. No credit will be given for late homeworks, late labs, or for projects turned in after 24 hours. Please do not beg and plead for exceptions. If some personal crisis disrupts your schedule one week, don't waste your time and ours by trying to fake it; just be sure you do the next week's work on time.

By the way, if you wait until the night before to do the homework or a project, you will probably experience some or all of the following: a shortage of available workstations, an unusually slow computer response, or a file server crash.

13 Lost and Found

When people bring me found items from lecture or lab, I take them to the Computer Science office, 387 Soda. Another place to check for lost items is the campus police office in Sproul Hall.

14 Questions and Answers

Q: Is it true that 61A is the weed-out course for wannabe CS majors?

A: No. The lower division sequence as a whole does determine admission to the major, but no one course is crucial. More to the point, the work in all of these courses is *not* designed to be especially hard; the upper division courses are much harder. The grading policy in 61A is not harsh and is *not curved* as it would be if we had weeding out in mind. However, you may take this course as an opportunity to weed *yourself* out; if you find that you don't enjoy the work, perhaps you aren't a computer scientist at heart.

Q: Why don't we learn some practical language like C++?

A: Firstly, Lisp *is* practical. Of the hundreds of languages that have been invented, Lisp is the second-oldest survivor, after Fortran. It hasn't lasted 35 years by being useless. Secondly, and more importantly, the goal of 61A isn't to teach you a language. The language is just the medium for the ideas in the course, and Lisp gets in the way less than most languages because it has very little syntax and because you don't have to worry about what's where in the computer memory. (Next semester you'll learn Java.) Finally, our textbook is **the best computer science book ever written**. It happens to use Lisp; if they'd used COBOL, we'd probably teach COBOL for the sake of this text.

Q: What's your advice on surviving this course?

A: Two things: Don't leave the homework and projects until the last minute, and **ask for help as soon as you don't understand something.**

Q: I am disabled and need special facilities or arrangements to do the course work. What should I do about it?

A: If you need special arrangements about class attendance, taking tests, etc., I'll be glad to accommodate you; please take the initiative about letting me know what you need. For example, if you want to take tests separately, that's fine, as long as you ensure that we've worked out the arrangements before the test. The Disabled Students Program (ext. 2-0518) has voice response terminals from which blind students can connect to our computers. **If English is not your native language,** and you have trouble understanding the course materials or lectures for that reason, please ask for help about that too.

Q: I don't like (or have a conflict with) my pre-assigned discussion section. Can I switch?

A: You must negotiate this with Jeff.

Q: What should we call you?

A: "Kurt" is just fine.

Q: I'm having trouble understanding the assignments. I've never had a problem like this in school before. Does this mean I'm not as good a programmer as I thought, or should I just wait a week or two and see if things clear up?

A: Neither. THIS COURSE IS CHALLENGING! In some ways, it might be the most challenging CS course you EVER take as an undergraduate. Most Berkeley students found high school pretty easy, and for many of you, this course will be the first real intellectual challenge you've met. You may have come to believe that everything should be easy for you. On the contrary; if you find your courses easy, you're taking the wrong courses! The whole reason you chose an excellent university was to stretch your mind. (If

you chose Berkeley for the sake of a prestigious diploma, maybe you should consider majoring in Business Administration.) *There is nothing shameful about asking for help.* You will learn a lot even if you do not get an A+. Every semester a few intelligent students end up in trouble in this course because they're too proud to come to office hours with questions. If you wait two weeks before you ask your question, by then you'll feel hopelessly behind, because the topics for those two weeks depend on the idea that you don't understand now.

Q: I have no prior programming experience, unlike those who have taken CS 3 that you regularly mention. Am I at a disadvantage to those students in terms of workload, grades, etc.?

A: Well, for the first couple weeks, youre definitely at a disadvantage. The cs3 students have already spent an entire semester learning scheme, higher-order procs, lambdas, recursion, and abstraction there is no reason why any of them should get less than perfect scores on any assignment from the first couple weeks. So, you will probably be spending more time and effort than they will for the first couple weeks and your grades over the first few assignments still (probably) wont be as good as theirs.

Fortunately, the class is not curved. It doesn't matter how well the cs3 students do; you need only be concerned with yourself. Many persons who have not taken cs3 get As in 61a. I havent seen the numbers myself, but I have heard that, statistically speaking, there is no difference between the average final grades of cs3 and non-cs3 students.

Q: I'm completely lost; I feel very awkward using scheme (I like my c++ much better) and I'm thinking about dropping the course. What do you think?

A: It's almost ironic that scheme is often harder to learn for people who have prior programming experience in other languages than for those who have never programmed before. Scheme requires a different way of thinking about problems and this can work against people who have had another, different sense of programming *per se* ingrained in them from the use of other languages.

Once you have become accustomed to it, however, you will begin thinking about problems in scheme-terms and feeling awkward coding in anything else. By the end of the course, scheme will be a tool that you use without even thinking about it (like writing with a pen). (Heidegger, anyone?)

How quickly you overcome your initial awkwardness with scheme is up to you: the more you play around with it, the faster you will become proficient. This class is really about thinking logically; if you are rational, reasonably intelligent, and willing to work very hard at absorbing new concepts, you will do very well in the course. If you fail to satisfy any of the three (especially the last), you will have a hard time.

If you do decide to stick it out, please be aware that the TAs and I are happy to help anyone who tries to help himself or herself. Dont be afraid to schedule office hours, etc. were here for you. Also, you may want to look into the recommended text 'Simply Scheme' by Brian Harvey. It is the book used in cs3.

15 First Assignments

Read section 1.1 of Abelson and Sussman as soon as possible. By Wednesday, read 1.3 of Abelson and Sussman. The first homework assignment is due next Sunday (check the reader or web site). You must log into your class account by Wednesday.