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

CS61A
Course Info

Kurt Meinz
Summer 2002

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 a 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!

This course expects some mathematical 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*. (For examples, go to `http://www-inst.eecs.berkeley.edu/~cs61a/misc/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 a disadvantage. You certainly have adequate background for this course if you are familiar with the idea of *recursion*: a procedure invoking itself as a subprocedure. 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 without spending 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 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 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.

In addition to the textbook, there is a reader containing necessary materials, including all assignments and material 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.

If you haven't used Unix before, you should also get the *User's Guide to Unix and the EECS Instructional Facilities* also available at CopyCentral.

We have also listed optional texts for the course. These really are optional! Don't just buy them because you see them on the shelf. One is the instructor's manual for the required text. It includes the authors' second thoughts about which ideas proved to be complicated and how to explain them, along with additional exercises. (It doesn't have solutions to exercises.) You may want this manual if you are excited by the course and want to get to know the authors and their ideas better, although it has been partly replaced by a web page about the book. The second optional text is *Simply Scheme*, by Harvey and Wright. This is sometimes 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. Most of you won't need these books.

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 web site.

The course reader includes the lecture notes. What this means is that you should be able to devote your effort during lecture to thinking, rather than to frantic scribbling.

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 between Monday and Tuesday's lectures, and again between Wednesday and Thursday's lecture. The discussion section meets between Tuesday and Wednesday's lecture, and again between Thursday and the next Monday's. You will also need to spend additional time working on the computers in the Soda Hall labs. Most weeks, the first meeting will be in our laboratory room, 271 Soda Hall; the second meeting will be in the classroom listed in the Schedule of Classes. Occasionally there may be two lab sessions and no classroom sessions. For example, **all meetings this week will be in the lab**.

An **updated list of discussion times and rooms is attached** and the most recent copy will always be on the course website; don't believe the printed schedule of classes or your enrollment form unless it agrees with this.

The discussion and lab sections are run by Teaching Assistants; each TA will handle enrollment and grading for his or her sections. We anticipate some rearrangements during the first week in response to oversubscribed or undersubscribed sections. **If you are waitlisted or your section has been cancelled,** you should communicate via email with the TA who is in charge of the sections that you would like to move into but be prepared to be flexible if your first choice is full. Please be in a definite discussion section by the end of this week, though, because much of the coursework will be done in groups of two to four students (the number depends on the activity); these groups will be set up by the TAs within each section.

You must have a computer account on the 61A course facility. You must set up your account *before Noon on Wednesday, June 26* 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. 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!

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.

Please do not sign up for a computer science course just to get a computer account, with the intention of dropping later. (Instead, come see a faculty member to discuss sponsorship of a non-class account for independent study, or you can get a free Unix account from the Open Computing Facility.) Accounts of students who are not doing the course work will be turned off by the second week of classes. Also, 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.

Students sometimes ask whether attendance is required or optional. Our expectation is that you will attend all class sessions, but you are adults and we will not police your attendance. However, if you take this to mean you can skip 6 weeks of section and then receive help from the TAs and instructor right before the final, you will be sorely mistaken. You will find that we are busy helping students that we have seen working hard all summer. Recognize that by not attending section or lecture, you are missing out on excellent opportunities to learn from the TAs and other students, as well as from the lecturer. Much of the learning in this course comes from lab activities, and later assignments *(including exam questions)* may build on those activities. Further, the TAs find it easier (and more enjoyable!) to help the students that they have gotten to know throughout the term. If you are missing school due to illness or some other emergency, inform your TA immediately.

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 pieces, 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. 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 easier, 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 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, we will try to encourage you to ask each other, and the TAs and myself for help.

6 Information Resources

Your first and most important resource for help in learning the material in this course is your fellow students. Your discussion section TA will assign you to a group of four students, and you will do many course activities with this group. You are responsible for helping each other learn.

The Teaching Assistants who teach the discussion sections are also available to answer questions. You may drop in during office hours, make appointments for other times, or communicate with them by electronic mail. Feel free to visit any of the TAs–not just your own! You may find that hearing different people's explanations helps you 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 particular TA or 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. To do this, subscribe to the `ucb.class.cs61a` newsgroup by saying

```
rn -q ucb.class.cs61a
```

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 class web page, with online versions of some of the documents we hand out:

```
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/
```

Tutoring services are provided by Eta Kappa Nu (HKN), the EECS honors society, and Upsilon Pi Epsilon, the Computer Science honors society. They share an office in 345 Soda; call them at 2-9952 or send

e-mail to `hkn@hkn` or to `upe@cory`.

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 be used from any EECS Instructional lab in Soda or Cory Hall.

The lab in 271 Soda Hall 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 floor 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. Since lab sections run from early morning until late evening, 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 readers include 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. Each homework assignment includes a suggested "feature of the assignment" for you to explore. These are entirely optional, and there is nothing to hand in about them. Do the real homework first, but if you have time, you will enjoy learning about the software tools available here.

The Computer Science Undergraduate Association (CSUA), Open Computing Facility (OCF), and Experimental Computing Facility (XCF) usually offer introductory Unix training sessions. Details will be announced when we have them.

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`

8 Computer Community Spirit

If you have lived in a dorm or other concentrated student housing, you have already learned that any facility shared by a large group of people is fertile ground for practical jokes. You've also learned that selfishness in the use of common facilities can lead to a lot of bad feeling. Computers are no different. For example, there is only a finite amount of file storage space. If you fill it up with digitized pictures of all your friends, other people can't get their homework done.

In the dorm, people generally have a good sense of perspective about what's funny and what isn't. Filling up your friend's room across the hall with balloons is funny. Filling it up with water balloons or live crickets or a 400 pound toilet is on the edge. Filling it up with epoxy isn't funny at all. But, for some reason, some people seem to lose that sense of perspective when it comes to computers. Perhaps it's because the damaged property is intangible; perhaps it's because with a computer you don't have to be physically near the victim. Whatever the reason, try to overcome it. It's not funny if someone can't complete the course work because

you deleted their files.

The operating system we use provides enough security so that nothing you do will mess up another user by accident if you're minding your own business. It is certainly possible to mess up the system deliberately. Many of you are familiar with the personal computer environment, in which some people consider it a mark of sophistication to write "virus" programs that interfere with other people's computers. You are now entering a different culture with different values. Our research work, as at any university, depends on collaboration both within our department and with colleagues elsewhere. Our computer systems are deliberately set up to *encourage* collaboration among their users, and that means encouraging easy access to one another's systems. This policy requires some degree of trust among the participants. If you've ever taken anything out of a safe deposit box at a bank, you know that it's possible to design a high-security shared facility, but that the cost is making it a big pain in the neck to use the secured data. Some computer systems are designed to have bank-level security, and everyone will think you're very clever if you figure out how to mess up such a system. Nobody will think you're clever if you mess up the 61A system.

The form you sign when you get your computer account says that it is for your use only and for course work only. We are not unreasonably strict in enforcing this rule. Nobody minds if you occasionally play a computer game late at night if it's the kind that doesn't wreck the keyboards or mice through repeated high-speed banging on one button. Nobody will object even if you occasionally bring a friend to play the game with you or if you write an occasional English paper on this facility instead of the official English Department computers. But if you are asked to give up the terminal by someone who wants to do course work and refuse, that's unacceptable. Remember, you and your fellow students are the ones who suffer from such obnoxiousness; the faculty and staff have other computers to work on.

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 projects early!

9 Network Etiquette

Our computer facility is part of a worldwide network that lets you communicate with other users both by electronic mail and by immediate connection if you're both logged on at the same time. You may find that the Internet, much like amateur radio, is a good way to make friends.

However, please remember that the network is *not* exactly like amateur radio, in that most of the people on our network are trying to get work done and don't want to spend time talking with you. Therefore, please do not send mail or `talk` requests to people whom you don't know. For example, if your best friend from home went to college somewhere else and you don't know his or her e-mail address, do not ask randomly chosen people at that college to locate your friend for you. (You can send mail to `postmaster` at most sites.)

The best way to get to know people on the net is to join newsgroups. The same program that you use for the class newsgroup will also let you subscribe to groups on an enormous range of topics, both technical and recreational. Most participants in these groups will welcome individual communication that's relevant to the newsgroup topic.

Here are a few rules of newsgroup etiquette: (1) Do not post to a group until you've read it for a couple of weeks, so you'll know what people consider appropriate topics for that group. (2) Do not post messages in which you quote all of someone else's long message and then add "Me too!" at the bottom. (3) Don't be sarcastic. If you're angry, wait until tomorrow to post your message. Remember, too, that the other person isn't necessarily just like you; he or she may be eight years old, or eighty. (4) **Do not** post, mail, or forward chain letters! You will certainly lose your Berkeley computer account and may find yourself under arrest for fraud.

It is strongly encouraged that you subscribe to the group `news.announce.newusers` for more information about posting to newsgroups.

10 Homework and Programming Assignments

Every week there will be problems assigned for you to work on, most 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. You are encouraged to do these exercises in groups of three or four students. They are NOT graded.
- **Homework assignments** consist mostly of exercises from the textbook; you'll do these whenever you can schedule time, either in the lab or at home. You may be accustomed to textbooks with huge numbers of boring, repetitive exercises. You won't find that in our text! Each assigned exercise teaches an important point.

  There are two homework assignments per week, but both are due on the Monday after they are assigned. These assignments are included in the course reader. (The first assignment is also attached to this handout.) You are encouraged to *discuss* the homework with other students. You should write up the solutions in groups of 2. Specific Homework requirements and grading policies are below.

  Some of the homework assignments include problems labeled as "Extra for Experts." These problems are entirely optional; do them only if you have finished the regular assignment and want to do something more challenging. There is no extra *credit* for these problems; people who need more credit shouldn't even be trying them, and people who are doing well in the course should be motivated by the desire to learn.

  You should try to complete the *reading* assignment for each week **before** the lecture. For example, you should read section 1.3 of the textbook by Wednesday. (Read section 1.1 as soon as possible this week!) You will have four class meetings (two lectures and two discussion/lab sections) to help you understand the assignment. 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.
- **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 them in groups. Specific Programming project requirements and grading policies are below.

Everything you turn in for grading must show your name(s), your computer account login(s), and your working group number for group assignments. Please cooperate about this; make sure they're visible on the *outside* of the paper you turn in, not buried in a comment in a listing.

11 Testing and Grading

If it were up to me, we wouldn't give grades at all. Since I can't do that, 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 students about their grades. To meet these goals, your course grade is computed using a point system with a total of 300 points:

```
3 midterms  3 * 40  120            15 homeworks    15 * 2  30
final               70             4 projects       4 * 20 80
```

There will be three midterms (tentatively set for the end of the third, fifth, and seventh weeks of the term) and a final. The midterms will be open book, open notes. (You may not use a computer during the exam.) In the past, some students have complained 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 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 general, midterms concentrate on the material that has been covered up to and including the day before the test. In this course, the later topics depend on the early ones, so you mustn't forget things after each test is over!

Each letter grade corresponds to a range of point scores: 280 points and up is an A+, 270–279 is A, and so on by steps of ten points to 170–179 points for a D−.

A+	280-300	A	270-279	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

If you make the effort to do the assigned work, you will do well on the weekly homework, since those points are awarded for effort and general understanding rather than for specific correct results. The projects do require correct solutions for full credit, but since the work is done in groups, if your group cooperates you're very likely to do well. Finally, the tests are meant to be easy for anyone who truly understands the material; they will require creative leaps, but leaps that you will be well-equipped to make if you have a solid foundation.

This grading formula implies that **there is no curve**; your grade will depend only on how well you (and, to a small extent, your partners) 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.)

If you believe we have misgraded an exam, return it to your TA with a note explaining your complaint. Only if you are unable to reach an agreement with the TA should you bring the test to me. The TA will carefully regrade *the entire test*, so be sure that your score will really improve through this regrading! By University policy, final exams may *not* be regraded. They 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.

12 Homework and Project Policies and Grading

Homework must be done in **groups of exactly 2**. That means, one assignment is turned in with both students names and logins on the first page. Additionally, **for the first 6 assignments, you must CHANGE your partner for successive assignments**. That means by the end of the 3rd week of class you have had 6 different partners! If you do not switch partners, you will receive no credit for that assignment. After the first 3 weeks, you may choose your own partners at your discretion.

The purpose of the homework is for you to learn the course, not to prove that you already know it. Therefore, the weekly homeworks are not graded on the correctness of your solutions, but on effort. You will get more credit for an entirely wrong answer that shows reasonable effort than for a correct solution that you copied from someone else. (But you should test your work! If your solution is incorrect, the grader will want to see some evidence that you know it's incorrect.)

Each homework is worth two points for a reasonable effort on all the problems, one point for some effort, or zero points for no effort.

Both of each week's homework assignments are due at 3:00 AM on the following Monday. There are two ways to turn in assignments: online and on paper. All homework assignments must be turned in online, and *may also optionally* be turned in on paper if you would like detailed comments on your work from your reader.

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.

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, still in that directory, give the shell command `submit hw5` (or whatever the assignment name is). We'll give more details in the lab.

Keep your graded papers until the semester is over. You may need them in case a grade is entered incorrectly.

The four programming projects *are* graded on correctness, as well as on your understanding of your solution. The first project is to be done individually, the second in a partnership of two, and the last two in groups of four. The last two projects are larger, and your entire group of four will work on a single solution, but the problems within each project are divided into two sets, and half of your group will work on each set.

The latter three group 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 your members' ability to answer correctly. Therefore, you must work together to ensure that all group members understand the entire project.

Your group will turn in *one copy* of each project, with all 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 3:00 AM on the second Monday after it is assigned (i.e. you have two weeks for each project), but there will be some exceptions. You'll get instructions about how to do this when the time comes.

13 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.

Unlike the homework and projects, 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. You will get negative credit for copied solutions, and repeated offenses will quickly lead to more severe penalties. 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 – 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 and your partner 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 your pair turns 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 group of 2 writes it up themselves. Working on the homework in groups is both a good way to learn and a lot more fun! Although the homework is graded on effort rather than on correctness, if you take the opportunity to discuss the homework with other students then you'll probably

solve every problem correctly.

Since the textbook exercises are largely the same from one semester to the next in this course, you may be tempted to turn the official published solutions collected by a friend who's already taken the course. Don't do it, for three reasons: First, it's dishonest. Second, the readers will recognize those solutions and you'll get caught. Third, *doing the homework is the main way you learn in this course.* Read the published solutions *after* you struggle with each problem yourself. Again, you are encouraged to talk about your solutions with others, but you should write it up in pairs, so that both members understand all the solutions to the problems. If you learned something from another pair, CITE IT, or else it is plagiarism.

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.

What does it mean to do an assignment as a group? The *best* groups solve each problem together, making sure that every member contributes to the discussion and that every member understands the group's ultimate solution. Your experience in this course will depend on the cooperation of your group more than anything else!

Second best is if you split up the problems so that each individual solves a few of them. This can be okay, as long as you then get together, after doing the individual work, to discuss the results and ensure that each member of the group understands every part of the project. It's best if your group also discusses the problems together *before* you split up to work on individual exercises, to make sure that everyone in the group understands the broad ideas of the assignment.

A bad group is one in which one group leader does all the work and the other members become spectators. Computer programming is a skill; you learn it by doing it. If you have a "freeloader" in your group, you're not doing him or her a favor! It's important that everyone be an active participant. Try to resolve any problems about working style within the group, but if that fails, ask me or your TA for help. As a last resort, if a member just won't cooperate, the group can "fire" that member *between projects* by notifying the TA, who will help you rearrange group memberships.

If you split up the work, then be *sure* that your group meets to collect the results before the last minute! If one group member fails to do the work, the entire group is responsible for ensuring that it gets finished. The ideal working arrangement is to meet early in the week to plan your tasks for the week, then get together a day or so later to confirm that everyone is mostly done and solve as a group any problems that the individual members can't solve, then meet for a third time *early* the day before it is due to collect everyone's work and solve any last-minute problems.

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.

14 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, 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.

15 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.

16 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: I am pre-enrolled for this course, and I'm planning to do the homework on my home computer. Do I still have to pick up a class account and log in by Wednesday to stay in the class?

A: Yes.

Q: I am a transfer student, and I'm pressed for time to fit in all my graduation requirements. I know how to program. Do I really have to take 61A?

A: Yes, unless you have taken this same course elsewhere. 61A is really very different from the usual first computer science course. However, your prior experience may well get you out of 61B, which is more nearly a standard second course. Mike Clancy is in charge of approving course equivalents.

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

A: First of all, 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. Second, and more important, 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 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 got the Nobel prize last year, and my uncle is Chancellor of Berkeley. Do I still have to use my class account by Wednesday Noon to stay in the class?

A: Yes.

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 the TA of the section you want to switch into. Please try to be settled into a definite section by the second week, when the group assignments will be made.

Q: Isn't it unfair that my grade depends in part on the performance of the other students in my group?

A: Do you complain about courses that are graded on a curve? It's very common to find a course in which your grade is *hurt* by someone else doing well in the course. If you can accept that, you should be much happier about an arrangement in which your grade is *helped* if you can help someone else learn.

In the worst case, you'd be doing it all yourself anyway– here, there's a lot of help so that you won't have to do that. But better still, you can get to know other people in the course; at some point, they will know something that you don't, and you'll have a better chance to make more friends.

Q: Can we form a group with students in other sections?

A: Generally not. One purpose of the scheduled lab meetings is to ensure that your entire group can spend some time working together with your TA available to help. If you want to be in the same group with a friend, arrange your schedules so that you can be in the same section. If there's some special reason why you think you should be an exception, negotiate with the TA or TAs involved.

Q: I'm thinking about buying a personal computer. What do you recommend?

A: For this course, and in general for computer science courses at Berkeley, you don't *need* a computer of your own at all; you can work in the labs on campus. If you just want to be able to connect to the campus computers from home, anything with a modem will do. (If you live in certain dorms, there is an Ethernet connection in your room, and having a computer with an Ethernet adaptor will be very handy.) If you want to work entirely within your home computer, you can get STk for PC-compatibles or Gambit for the Macintosh in 387 Soda.

Some of our students, especially the ones with a particular interest in system administration, choose to run one of the free versions of Unix at home, usually Linux or FreeBSD, but to each their own. Learning to use some flavor of UNIX takes more effort than using commercial systems, but you learn a lot in the process.

Q: One of the other people in my working group never does any work. What should we do about it?

A: First of all, try to find out why. Sometimes people give up because they're having trouble understanding something. If that's the problem, see if you can teach your partner and get him or her back on track. Also, try to find out what his or her *strengths* are—how he or she can best contribute to the group's efforts. But sometimes people get distracted from coursework for non-academic reasons. If you can't resolve the problem within the group, talk with your TA. With your TA's permission, your group may fire a member *between* projects (not during a project). Your TA will generally allow you to fire members who make no effort to cooperate, but not ones who are trying but having difficulty in the course. (If someone in your group insists on doing all the work, that also counts as not cooperating.)

Firing a group member is a last resort. On the other hand, if you do have a problem with someone in your group, you should be sure to resolve it quickly, because we will not accept hard-luck stories at the end of the semester about how you lost points undeservedly because the other people in your group never did their share of the work.

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.

17 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 Monday (check the reader or web site). You must log into your class account by Wednesday.