# CS 61A: SICP—General Course Information

Instructor: Kevin Lin

June 26, 2006

## 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, not about any particular programming language. We consider a series of techniques for controlling program complexity, such as functional programming, data abstraction, object-oriented programming, and query systems. To get past generalities you must have programming practice in some particular language, and in this course we 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?

Math 1A is a corequisite for 61A. (That is, it may be taken concurrently.)

In the past we did not impose any programming-related prerequisites for admission to 61A. However, in recent years 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. If you've taken the CS Advanced Placement AB course in Pascal or C++, you are certainly ready for 61A. There may be a short placement quiz the evening of the first lecture, for those who do not

have an approved earlier course credit, in which you are asked to write a recursive procedure in any language. (We usually don't bother with the quiz, but if you don't think you could pass such a quiz, you might still want to defer 61A.)

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.

Non-technical students (for instance, ones who prefer Math 16A to Math 1A) should probably avoid CS 61A entirely, and should take CS 3 or 3S to satisfy their need for or interest in a programming 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 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!!!**

Most semesters we have a reader that contains the homework assignments, lab assignments, and projects. This semester, however, I will be posting all such assignments online on the course website (`http://inst.eecs.berkeley.edu/~cs61a`). I will also be posting my lecture notes and handouts online.

If you haven't used Unix before, you should definitely start learning about it as soon as possible. The main course webpage will contain a link to a brief introduction to Unix and the computing resources at Berkeley.

We have 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 the textbook for CS 3; it gives a slower and gentler introduction to the first few 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. You can get a CD-ROM with these materials at the CS front desk, 387 Soda, at no charge. You can also find up-to-the-minute downloadable copies of this distribution at `http://inst.EECS.Berkeley.EDU/~instcd`.

# 4 Enrollment—Laboratory and Discussion Sections

In addition to the lectures Monday, Tuesday, Wednesday, and Thursday, the course consists of two discussion sections and two lab sections per week. (You will probably also use some additional unscheduled lab time to do homework problems.) Lab sections will be in 271 Soda Hall on Mondays and Wednesdays; discussion sections will be in 310 Soda Hall on Tuesdays and Thursdays. However, for the first week, we will have labs on Monday, Tuesday, and Wednesday, and discussion on Thursday only.

The discussion sections are run by Teaching Assistants; each TA will handle enrollment for his or her sections. We anticipate some rearrangements during the second week in response to oversubscribed or undersubscribed sections. If you are not pre-enrolled, you should pick a discussion section, but be prepared to shift if your first choice is full. Please try to be in a definite discussion section by the second 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 *this week* because that is how we know who is really in the class. If you are pre-enrolled but do not set up your account this week, you may be dropped from the course. Account forms will be distributed at the first lab. If you missed the first lecture, see Sue DeVries or Cindy Palwick in 385 Soda Hall to get your account. The first time you log in, you will be asked to type in your name and student ID number, if you have one. Please follow the instructions carefully. You must get your account *and log into it* no later than **4pm Friday** so that we have an accurate class count. (In any case, you should be starting on the first homework assignment by then.)

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 except to submit your assignments.

You may also wish to use your class account remotely. You're on your own in figuring out how to do this, but instructions for Windows computers will be available at `http://inst.eecs.berkeley.edu/connecting.html`. If you're using a Mac you'll probably have to install X11.

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 or the Computer Science Undergraduate Association.) Accounts of students who are not doing the course work will be turned off by the third 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.

If you are not pre-enrolled and want to take 61A, you have to add the course using Tele-BEARS. If you are something other than a regular Berkeley undergraduate, then you probably need a signature on a form admitting you to the course.

Students sometimes ask whether section attendance is required or optional. Our expectation is that you will attend all class sessions, but you are adults and we are not going to police your attendance. However, much of the learning in this course comes from lab activities, and later assignments (including exam questions) may build on those activities. It is up to you to find out what happened at any class session that you miss.

# 5 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 two students. You are responsible for helping each other learn.

The class might have a staff of undergraduate Lab Assistants (LAs). Each LA will have scheduled hours to be in the lab. Whenever an LA is in the lab (during scheduled hours or not) you may request that s/he answer questions about the homework or programs (but *not* do them for you).

The Instructor and 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 us by electronic mail.

For technical questions about the homework or about the computer facility, or administrative questions such as missing homework grades, send electronic mail to your particular TA or reader.

There is an electronic bulletin board system that you can use to communicate with other 61A students. To do this, subscribe to the `ucb.class.cs61a` newsgroup using any news reader, such as pine, trn, Netscape, or Outlook Express; the Berkeley news server is `news.berkeley.edu`. If you have no idea what any of this means, don't worry, your TA will explain it to you.

There are also useful links on the course webpage.

# 6 Computer Community Spirit

If you live 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 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 may be funny if your friend sees an unexpected message when s/he logs in, but it's not funny if s/he can't complete the course work because of deleted 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 selfishness; the faculty and staff have other computers to work on.

# 7 Homework and Programming Assignments

Each week there will be problems assigned for you to work on, most of which will involve writing and debugging computer programs. You'll work on some of these problems individually, and some in groups. These assignments come in three categories:

**Laboratory exercises** are short, relatively simple exercises designed to introduce a new topic. You should do these in groups. Labs are not worth any points—this is because the TA's time in lab is better spent helping students understand the lab than running around checking students off. But we still think that they're an essential part of the course and that you shouldn't skip them.

**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. These assignments will be posted online. I will assign two homework assignments each week. Each homework assignment will be due the Wednesday of the *following* week that it was assigned. You are encouraged to *discuss* the homework with other students, but each of you must prepare and turn in your own solutions. (More on this later.)

**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 semester. The first two projects will be done individually; the last two in groups of two students.

Some of the weekly assignments include problems labelled 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.

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 full credit for an entirely wrong answer that shows reasonable effort!** (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 one point for a reasonable effort, zero points for a missing homework or one that seems to show no effort, or **negative four ($-4$) points** for a solution copied from someone else.

The four programming **projects** *are* graded on correctness, as well as on your understanding of your solution. The first two projects will be done individually; the last two in groups of two students, but the work is split up so that each problem is done by one student. You must work together to ensure that both group members understand the complete results. Projects must be turned in both online and on paper (see below).

Copying someone else's work does not count as "reasonable effort"! This includes copying from your friend who took the course last semester as well as copying from other current students. You will get negative credit for copied solutions, and repeated offenses will 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. **DON'T DO IT**, you will be caught.

There are reading assignments for each lecture, which will be posted online. You should try to complete the *reading* assignment **before** the lecture.

Each homework assignment will be turned in by 11:59 PM Wednesday of the following week. The homework assignments *must* be turned in online (your TA will show you how to do this), and *may also* 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. **Keep your graded papers** until the semester is over. You may need them in case a grade is entered incorrectly.

**Online turnin:** You must create a directory (you'll learn how to do that in the lab) with the official assignment name, which will be something like `hw5` or `proj2`. Put in that directory all files 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.

The programming projects must be turned in online and in paper; the deadlines will be posted online. For the group projects, only one member of the group will submit the entire project for the group.

**Everything you turn in on paper must show your name(s), your computer account(s), and your section number.** 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. For online submissions, your name(s), computer account(s), and section number must be included in a comment at the beginning of the file.

# 8 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 encourage you to do the course work and reward reasonable effort with reasonable grades; 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 and a final. Dates, times, and locations will be posted online. My goal will be to write one-hour tests, but you'll have two hours to work on them. The relatively large number of tests should reduce your anxiety about ruining your grade by misunderstanding any one question. In general, tests concentrate on the material that has been covered up to and including the week 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. Finally, the tests are meant to be easy for anyone who understands the material; they do not require great creative leaps.

This grading formula implies that **there is no curve**; your grade will depend only on how well you (and, to a small extent, your group 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, first be sure that you understand the solutions and grading standards that we'll post online soon after the exam. If your paper was misgraded *according to those standards*, 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.

# 9 Cooperative Learning Policy

With the obvious exception of exams, we *encourage* you to discuss *all* of the course activities with your friends as you are working on them.

Each student must solve the weekly homework assignments individually; **it is by struggling with the homework problems that you learn the course material!** However, before you develop your own solution to each problem you are encouraged to discuss it with other students, in groups as large or small as you like. **When you turn in your solution, you must give credit to any other student(s) who contributed to your work.** 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.

Similarly, each student should solve each problem in the schedule lab activities, but you are welcome to discuss your efforts with your neighbors in the lab.

The first two programming projects will be done individually. The last two projects are larger, and will be done in groups of two, but with each individual student responsible for specific problems within the project. The groups will be chosen by the TA of your discussion

section, although we'll consider requests for specific partners or unusual constraints. Your group will turn in *one copy* of each project, with both of your names and logins listed on it.

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 your project partner and your TA, so that nobody is depending on you to do something you can't finish.

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. Reading the homework solutions each week is an excellent way to prepare for the exams.**

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." 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.

In my experience, nobody begins the semester with the intention of cheating. 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 in a drunken orgy instead of doing your homework, please overcome your guilt feelings and **ask for help as soon as you need it.**

# 10 Lateness

A programming project that is not ready by the deadline may be turned in until 24 hours after the deadline. These late projects will count for 1/2 of the earned score. No credit will be given for late homeworks, or for projects that are more than 24 hours late. 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.

# 11 What Should We Call You?

Just call me Kevin. I'm not a Professor, I'm just an Instructor for the Summer. If you try to call me something like "Professor Lin", I probably won't realize that you're talking to me. Don't call me "Mr. Lin"; we're not in grade school!