CS 61B: Data Structures (Spring 2002)
Course Overview

Prof. Jonathan Shewchuk

1 Introduction

CS 61B is the second course in the CS 61 series. In this course, you will study advanced programming techniques including data structures, abstract data types, interfaces, and algorithms for sorting and searching, and you will get a taste of "software engineering"—the design and implementation of large programs.

Please read this document carefully. It contains answers to most of the questions that students ask during the first few weeks of class. The subjects include: how to contact the staff, prerequisites, textbooks, labs, grading, late penalties, and policies on academic misconduct. There is a class Web page at `http://www.cs.berkeley.edu/~jrs/61b`. A list of discussion sections, labs, and the TAs who run them is linked from the Web page. A tentative syllabus, which includes reading assignments, exam dates, and due dates, is also available. Several online handouts are available. Please check the class Web page at the beginning of the semester and regularly throughout to learn about new information and syllabus changes.

If you have a general question about something *not* covered herein, the best option is to post a message in the `ucb.class.cs61b` newsgroup. We (the instructor and TAs) check the newsgroup regularly, and other students will be able to help you too. Other students will also be able to benefit from the answers. If you don't want to make your question public, you may send email to `cs61b@cory.eecs`. Your email will be forwarded to the instructor and all the TAs. If you wish to talk with one of us individually, you are welcome to come to our office hours, posted on our doors and linked from the Web page. If the office hours are not convenient, you may make an appointment with any of us by email. There are about 40 of you to every one of us, so please reserve email for the questions you can't get answered in office hours, in discussion sections, or through the class newsgroup.

Instructor:
Prof. Jonathan Shewchuk
Office: 625 Soda Hall
Phone: 642-3936
Email: `jrs@cory.eecs`

Course Secretary:
Sue Devries
Office: 385 Soda Hall
Phone: 642-0930
Email: `csoffice@cs`

Teaching Assistants:
Anthony Giardullo, cs61b-ta@cory.eecs
Barath Raghavan, barath@cory.eecs
Ravi Kolluri, cs61b-tc@cory.eecs
Adam Kirk, akirk@cs
Steve Sinha, cs61b-te@cory.eecs
Yosen Lin, yosenl@cory.eecs
Steve Wong, cs61b-tg@cory.eecs
Neha Kumar, cs61b-th@cory.eecs

Lectures are Mondays, Wednesdays, and Fridays from 2:00 pm to 3:00 pm in 1 Pimental Hall.

2 Prerequisites

The prerequisite for CS 61B is CS 61A or Engineering 77N. CS 61A is an introductory course taught in Scheme, and covers topics like recursion, higher order functions, object-oriented programming, interpreters, simple data structures (lists and tables), and the basics of using the computers in our labs. E 77N covers the same topics, but is taught in Matlab. Although TeleBears does not prevent someone from registering for the course without prerequisites, checking software is run from time to time to automatically drop such students from the class. See the drop list outside of 379 Soda if you think this might happen to you.

If you have not taken either CS 61A or E 77N, chances are you will not be admitted to the course. If you have taken a course you feel is very similar to CS 61A (not just a programming introduction, but one that covers the topics above), fill out an appeal form in the main CS office, 387 Soda Hall. I do not handle appeals, so please do not attempt to lobby me for admission to the course. If you try to stay in the course when you have not satisfied the prerequisite, you will receive an F as your final grade.

We will be using the Java programming language in this course. Knowledge of Java (or related languages like C++) is *not* required for this course. You may have heard that Java is a language for programming Web applications and developing animated home pages. This is partly true, but neither of these will be covered in this course. We will simply use Java as a better language for writing large, modular programs than many of the alternatives.

If you have already taken a data structures course in any programming language (e.g., Pascal), you may not need to take CS 61B. If you know Java as well, you might be able to skip CS 61B entirely, and if you don't, you might only need to take CS 9G (self-paced Java). If you feel that this course may be repeating prior experience, please see Mike Clancy in 779 Soda.

If you are not familiar with the Unix operating system and basic tools, either because you did not take CS 61A, or you got through CS 61A without understanding Unix, it is important that you learn. Some student groups, including CSUA, teach help sessions on Unix.

3 Textbooks

There are two required textbooks for the course.

- David Arnow and Gerald Weiss, *Introduction to Programming Using Java: An Object-Oriented Approach*, Java 2 Update, Addison-Wesley, 2000. ISBN # 0-201-75159-3 or ISBN # 0-201-61272-0.
- Michael T. Goodrich and Roberto Tamassia, *Data Structures and Algorithms in Java*, Second Edition, John Wiley & Sons, 2001. ISBN # 0-471-38367-8.

These texts should be available at the ASUC bookstore or across Bancroft at either Ned's or the Campus Textbook Exchange. The first edition of Goodrich and Tamassia (ISBN # 0-471-19308-9) will serve you just as well as the new one. In fact, we will cover in class several sections that are in the first edition and not the second; copies of these sections will be provided to everyone who has the second edition. Unfortunately, the sections are numbered completely differently in the first and second editions. You can probably get away with the first edition of Arnow and Weiss (whose cover lacks the red "Java 2 Update" box), but it includes the occasional trace of Java 1 code that might not work correctly with Java 2. The Java 2 Update edition is preferable.

Arnow and Weiss give an introduction to programming in Java. We will be using the Arnow and Weiss book during the first month, so purchase it soon. Their book is not meant to be a complete reference to all of the concepts in Java. If you are an experienced C hacker, I also recommend David Flanagan, *Java in a Nutshell*. The chapter entitled *How Java Differs from C* can bring a C programmer up to speed in Java remarkably quickly. The Goodrich and Tamassia book is about data structures and uses Java in the code examples.

You should also buy the class reader, available at Copy Central at 2483 Hearst. The bulk of the reader is old CS 61B exams and sample solutions. The remainder is information on using the compilers, debugger, and editor.

Electronic copies of all class handouts will also appear on the Web page. There may be up to three types of files. Raw ASCII text (`README` files and other filenames without an extension) should be printed using the `enscript` command. PostScript files (filenames ending in `.ps`) must be printed using the `lpr` command. (Do **not** use `enscript` on a `.ps` file! You'll just print reams of garbage.) HTML files (filenames ending in `.html`) should be printed using the Print command on the File menu in Netscape.

The TAs and I will post announcements, clarifications, hints, and other information in the `ucb.class.cs61b` newsgroup, which you should read regularly whether you post questions to it or not, so that you're not the last to find out about major changes to assignments and midterm dates. Send questions or information of general class-related interest to this newsgroup using the Netscape Web browser or a news reader like `trn` or `rn`, which have facilities for reading and posting messages.

4 Laboratory and Discussion Sections

In addition to the lectures, you will attend a discussion section for one hour each week and a scheduled lab for two hours each week. The discussion sections meet between the Monday and Wednesday lectures, whereas the labs meet between the Wednesday and Friday lectures. All labs meet in 275 Soda. You should have the same TA and student colleagues in your lab and discussion sections. Because of limited space in

the labs, you are only allowed to attend a lab in which you are officially enrolled through TeleBears. If you are on a waiting list, attend a lab that TeleBears claims has open space. For discussion sections, we are not that strict, although you are strongly encouraged to attend the section for which you are registered. Your Lab TA will be your "primary" TA—he or she will check off your lab assignments, return your midterms, and help with grading corrections. You should feel free to attend any of the staff office hours (not just your own TA's) and ask any of us for help.

Laboratory sections, which are two-hour slots in the second half of the week, are mandatory. Each week you will solve an assignment in the lab, and have points checked off by your TA. Discussions sections are not mandatory, and nothing done in section will directly affect your grade. On the other hand, discussion sections are your best opportunity to ask questions and learn interactively, and some examples worked out in section will be helpful on the graded assignments. Midterms will be returned in section.

Account forms will be given out in lab during the first week, so it is important that you attend. If you miss your first lab, see your TA as soon as possible. *It is important that you login to your account and change your password during the first week;* we use the information about who has logged in along with the TeleBears roster to determine who will receive grades in the class.

The Soda labs are open from 7:00 am to 6:30 pm Monday through Friday. Outside these hours, the doors to the building and lab are locked. You will need to obtain a card key for after hours access; these will not be available immediately, because we need an accurate list of who is in the class first. An announcement will be made in lecture when they are ready. For those of you with labs in the late evening, someone from your lab will wait near the third floor entrance of Soda to let you into the building.

5 Enrollment

Make sure that you are enrolled in a lab/discussion section that has space for you. To find a lab that has space, see the online class schedule for laboratory enrollment levels.

If you are on the waiting list for the course, the reason is that you are waiting to be admitted to a lab/discussion section that is currently full. You need to choose a section that is not full if you want to take the class. We will try to add enough sections so that everyone with the prerequisites is admitted to the course (subject to TA availability), but the new lab times will be early in the morning or late in the evening. We cannot let you into sections that are full, so if you insist on waiting for a full section to have space, you will not be allowed to attend lab (and therefore you won't get credit for the lab), and you will most likely never be admitted to the course.

If you are something other than a regular Berkeley undergraduate, then you probably need a signature on a form admitting you to the course. I cannot promise to admit those of you who are not regular Berkeley students. In particular, I might not sign any concurrent enrollment or UC Extension forms until after the *second* week of classes. Meanwhile, you should get your computer account and begin doing the course work on the assumption that you will be admitted. You will need to choose one of the lab times that is not overbooked.

6 Miss a Lecture? Take Lousy Notes?

For those of you who miss lectures, videos of classes past will be available through `http://webcast.berkeley.edu`. Live lectures may be available through this URL as well. At my convenience, I might also post lecture notes on the class Web site within a day or two after each lecture.

7 Course Work and Grading

There are a total of 200 points you can earn toward your final grade in the course. There will be two midterm exams, each worth 25 points, and a final exam worth 50 points. The test coverage is cumulative, so material from the beginning of the course may be tested in either midterm or the final exam. All exams will be graded by the TAs and professor.

In addition to exams, there are three types of assignments: homeworks, labs, and projects. Homeworks (roughly one per week) involve written assignments and programming. Homeworks must be done individually. They are typically due before lecture on Wednesdays, and will be graded by one of the class readers. Homeworks are worth a total of 20 points out of the 200, and each homework is equally weighted. Your worst homework grade will be dropped. For example, if there are 11 homeworks during the semester, each of the 10 best is worth 2 points of your final grade. You will find that some homeworks are much easier than others, but they all have the same weight.

Labs are short programming assignments that must be done during the scheduled lab period in the latter part of the week. Labs are done in teams of two. Grading of labs is done by having certain steps checked off by your TA or a lab assistant. Labs are worth 10 points of your final grade, and each lab is equally weighted. Since I expect you to have conflicts from time to time during the semester, we will ignore your two lowest lab grades.

The remaining 70 points of your final grade will come from the programming projects. There will be three of these during the semester. The first and last projects will be worth 20 points; the second will be worth 30 points. You will do the first project individually, and the second and third projects in teams of two or three students each.

The projects are a great deal of work, and cannot be put off until the last moment. If you start working on a project a few days before its due date, you will not be finished by the deadline. Not even close.

Your final letter grade will be determined by the following formula: 185 or above is an A+, 175–184 is an A, and down by ten points each to the lowest passing grade (D–, 75–84). In other words, *there is no curve*; your grade will depend only on how you do and not on how well everyone else does. Occasionaly, we *may* decide that one exam was unreasonably difficult, and raise the scores on that assignment. However, our experience is that grades on homeworks and projects are higher than on exams, so you should assume this will be the case for your own grades and not be surprised if your exam grades are lower than your final grade, while your homeworks and projects are higher.

Most, and perhaps all, assignments will be turned in electronically. Your grades will be recorded online and can be viewed using the `glookup` program.

If you believe we have misgraded a midterm exam question or project, return it to your lab TA with a *written note on a separate piece of paper* explaining the problem. If you're requesting an exam regrade, please staple this paper to the *front* of the exam. (*Not* the back! Please!) The entire exam or project may be regraded, so be sure to check the solutions to confirm that your final grade will go up after regrading. All requests for regrades and recording corrections must be made within two weeks after you receive the graded assignment or exam. By University policy, final exams may *not* be regraded.

A course grade of Incomplete 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.

7.1 Lateness

We will give no credit for written homework or labs turned in after the deadline, so that you can discuss the solutions in your discussion sections. Please do not ask for extensions for homework or labs—each of these assignments is worth very few points. Late homework and labs will not be accepted for any reason whatsoever, even emergencies. We drop your lowest homework grade and your two lowest lab grades to give you leeway to handle personal crises; we don't feel obliged to offer more than that.

On programming projects, we do allow assignments to be turned in late, but there is a penalty. If a project is N hours late, we'll reduce your earned score by $\lceil N/2 \rceil$ percent. While this gives you some leeway for putting the final touches on a project, do not stretch the deadlines too far. A project that is one day late will lose 12% of your earned score. After five days, even a perfect solution won't earn a passing grade.

7.2 Exams

The two midterms are scheduled to be held in lecture on Monday, February 25 and Monday, April 15. CS 61B is in Final Exam Group 20, for which the final exam is scheduled for Friday, May 24, from 5:00 to 8:00 pm. The location of the final will be announced by the University later in the semester.

8 Policy on Collaboration and Cheating

Cheating on a homework, lab, or project will earn you the maximum negative grade on that assignment. For example, if you cheat on a project worth 20 points, your grade on that project will be -20. Cheating on an exam, or cheating twice in any way, will earn you an F in the course. I reserve the right to assign an F in the course to anyone who cheats on a project once, though I will probably not exercise this right. All incidents of cheating will be reported to the Office of Student Conduct, who will maintain records of your academic misconduct throughout your undergraduate career.

We encourage you to help each other learn the material by discussing the work *before* you do each assignment. I believe that most students can distinguish between helping other students and cheating. For example, explaining the meaning of a question or offering advice on what a compiler error message means are interactions that we encourage. On the other hand, you should **never** have another student's solution in your possession, either electronically or on paper. (We will call this the "No Code Rule.") If you are not sure whether a particular interaction is appropriate, talk to your TA or the instructor. If you receive a significant idea from someone in the class, explicitly acknowledge that person in your solution. Not only is this a good scholarly conduct, it also protects you from accusations of theft of your colleagues' ideas.

Presenting another person's work as your own constitutes cheating, whether that person is a friend, an unknown student in this class or a previous semester's class, or an anonymous programmer on the Web who happens to have solved the problem you've been asked to solve. Everything you turn in must be your own doing, and it is your responsibility to make it clear to the graders that it really is your own work. The following activities are specifically forbidden in all graded course work:

- Possession (or theft) of another student's solution or partial solution in any form (electronic, hand-written, or printed).
- Giving a solution or partial solution to another student, even with the explicit understanding that it will not be copied.

- Working together to develop a single solution and then turning in copies (or modified versions) of that solution under multiple names.

You will do some of the projects in teams of two or three students. Any assignment that is not designated as a team assignment must be done individually. On team assignments, the rules for individuals given above apply to teams. You may not work with another team or share solutions across teams. Each individual in a team is responsible for the entire project, which means that you will be held responsible if your partner uses another group's solution to produce part of your team's solution. Once you've begun coding a project, you may not change the size of your team or exchange partners without our permission. If your team has irreconcilable conflicts after beginning a project, you must speak to me before breaking up or reforming your team. Only one of the new teams (at most) will be allowed to keep the code developed thus far.

Cheating will be policed by advanced cheating-detection software. If you share code with another team, you will be caught, even if you take steps to hide your cheating.

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. Some students get into this situation because they are afraid of an unpleasant conversation with a professor if they admit to not understanding something. I would much rather deal with your misunderstanding early than deal with its consequences later. Even if you are convinced that you are the only person in the class that doesn't understand the material, and that it is entirely your fault for having fallen behind, please overcome your feeling of guilt and ask for help as soon as you need it. Remember that the other students in the class are working under similar constraints—they are taking multiple classes and are often holding down outside employment.