CS 61B: Data Structures (Spring 2013)
Course Overview

Instructor:
Prof. Jonathan Shewchuk
Office: 529 Soda Hall
Phone: 642-3936
Email: jrs@cory.eecs

Teaching Assistants:
Akihiro Matsukawa, cs61b-ta@cory.eecs
Bryan Mau, cs61b-tb@cory.eecs
Lu Cheng, cs61b-tc@cory.eecs
Daniel Wang, cs61b-td@cory.eecs
Yun Seong Park, cs61b-te@cory.eecs
Mona Kumari Gupta, cs61b-tf@cory.eecs
Bharathkumar Gunasekaran, cs61b-tg@cory.eecs
Harrison Wallace, cs61b-th@cory.eecs
Rohan Dilip Salantry, cs61b-ti@cory.eecs
Jeff Chang, cs61b-tj@cory.eecs

Lectures are Mondays, Wednesdays, and Fridays from 5:30 pm to 6:30 pm in 155 Dwinelle Hall. The class web page is at http://www.cs.berkeley.edu/~jrs/61b.

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, encapsulation, 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.

Please check the class web page at the beginning of the semester and regularly throughout. A list of discussion sections, labs, and the TAs who run them is linked from the web page. A tentative syllabus,

which includes lecture topics, exam dates, and homework due dates, is also available there. Several online handouts are available. Finally, your reading assignments are listed there. Please keep up with them. There will not be reminders in class.

If you have a general question about something *not* covered herein, the best option is to post a message on the CS 61B Piazza discussion board. The Piazza board is **required reading**, whether you post questions to it or not. The TAs and I will post announcements, clarifications, hints, and other information there. If you don't read it, you may be the last to find out about review sessions and major changes to assignments and due dates. We (the instructor and TAs) check the board 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, or if your question would expose part of the answer to a homework assignment, you may send email to `cs61b@cory.eecs`. Your email will be forwarded to the instructor and all the TAs. You are always 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 50 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 Piazza board.

2 Prerequisites

The prerequisite for CS 61B is CS 61A or Engineering 7. CS 61A is an introductory course taught in Python or 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 7 covers the same topics, but is taught in Matlab. We assume that you have mastered recursion before taking 61B; if you do not have extensive practice with it, you will have difficulty with this class. Although TeleBears does not prevent someone from registering for the course without prerequisites, I won't promise that you won't be dropped from the class later.

We will be using the Java programming language in this course. Knowledge of Java is *not* required for this course.

If you have already taken a data structures course in any programming language (e.g., C), you may not need to take CS 61B. If you know Java too, you might be able to skip CS 61B entirely, and if you don't, you might only need to take CS 47B or CS 9G. If you feel that this course may be repeating prior experience, please see Brian Harvey in 781 Soda Hall.

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 textbooks for the course.

- Required: Kathy Sierra and Bert Bates, *Head First Java*, Second Edition, O'Reilly, 2005. ISBN # 0-596-00920-8.
- Optional: Michael T. Goodrich and Roberto Tamassia, *Data Structures and Algorithms in Java*, Fifth Edition, John Wiley & Sons, 2010. ISBN # 0-470-38326-7.

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 Sierra and Bates will serve you just as well as the new one. Likewise, the first, third, or fourth edition of Goodrich and Tamassia will serve you just as well as the new one. The second edition will serve almost as well, but it is missing sections that we will cover on splay trees and disjoint sets; the first, third, fourth, and fifth editions have these sections. Unfortunately, the sections and pages are numbered differently in all the editions of both books, so the reading assignments on the class web page apply only to the newest editions. I have taught this course from every edition of both books, so if you have an old edition, you can find the correct readings by going to my home page and finding the links for old offerings of this course.

Sierra and Bates give an introduction to programming in Java. We will be using the Sierra book from the beginning, so purchase it soon. It is an excellent book, and Amazon sells it pretty cheaply. 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. (That's how I learned Java.)

Goodrich and Tamassia give an introduction to data structures. Although I required this book in the past, I am now listing it as "optional" because it is expensive and some students do fine reading just my lecture notes. But I still recommend buying and reading it, as it discusses data structures in more detail and from a different point of view than I do.

You should also buy the class reader, available at Vick Copy at 1879 Euclid, near Hearst. The bulk of the reader is old CS 61B exams, which *will not be provided online* (though the solutions are). By far the best way to study for the exams is to try the old ones. The remainder of the reader is information on using the compiler, debugger, and editor.

Electronic copies of all class handouts will also appear on the web page. There may be up to four 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`) can be viewed using Ghostview (`gv`), and can be printed using the `lpr` command. PDF files (filenames ending in `.pdf`) can be viewed and printed using `acroread` (Acrobat Reader). Do **not** use `enscript` on a `.ps` or `.pdf` file! You'll just print reams of garbage. HTML files (filenames ending in `.html`) can be viewed and printed from Firefox, Mozilla, Netscape, Microsoft Explorer, Chrome, etc.

4 Laboratory and Discussion Sections

In addition to the lectures, you will attend a discussion section for one hour each week and a lab for two hours each week. The labs meet between the Monday and Wednesday lectures, and the discussion sections

meet between the Wednesday and Friday lectures. Labs meet in 273 or 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 the lab in which you are officially enrolled through TeleBears. If you are not enrolled in any lab (e.g. you're on a waiting list or you're a concurrent enrollment student), attend a lab that has room to accommodate you. (You might have to try several labs to find one whose TA has room for you, so don't wait until the last lab of the week.) For discussion sections, we are not strict, although you are 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. However, 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 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, but many ideas will be discussed in section that don't come up in lecture, and a few of these ideas will appear in the exams and homeworks. Discussion sections are also your best opportunity to ask questions and learn interactively. Midterms will be returned in section.

Account forms will be given out in lab during the first lab, so it is important that you attend. If you miss your first lab, see a TA as soon as you can. *It is important that you login to your account and change your password as soon as possible;* it takes a day or so after you login for the first time before the system will enable your ability to submit homeworks.

The Soda labs are open from 7:00 am to 6:30 pm Monday through Friday. Outside these hours (and on weekends and holidays), the doors to the building and the elevators are locked, and you will need a keycard to enter. If you are a Berkeley student, your student ID card serves as a keycard, but you must apply to have it enabled at 387 Soda or 391 Cory. If you are a concurrent enrollment student, you must pay $20 for a keycard ($5 non-refundable fee + $15 refundable deposit).

5 Enrollment

This is a somewhat disastrous semester for CS 61B, because the university allocated us a too-small room and the fire code legally prevents us from enrolling more than 110% of the number of seats in 155 Dwinelle. Roughly 150 students not majoring in computer science were thrown off the waiting list, and about 80 majors are still on the waiting list. One student on the waiting list will be admitted for each enrolled student who drops the class; I estimate that will be 40–50 students (but I might be way off).

To be admitted to the class, you must be waiting for a lab/discussion section in which a space opens up. If you see a lab section with an open space at a time you can attend, switch to it as fast as you can, even if it's not the time you would prefer. Until then, find and attend a lab whose TA thinks there is enough room to accommodate you.

If you are something other than a regular Berkeley undergraduate, you need me to approve your concurrent enrollment application online. Unfortunately, that will only happen if the entire waiting list of Berkeley students clears out before the enrollment deadline. This is extremely unlikely, but if you want to take a shot, find and attend a lab that has room to accommodate you, and send me an email to let me know your application is pending.

6 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) are short programs. Homeworks must be done individually. They are due before lecture on Fridays, 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.

Labs are short programming assignments that must be done during your scheduled lab period. 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 some of you to have conflicts during the semester, we will drop your lowest two 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 chart.

Points	**Grade**
185–200	A+
175–184.99	A
165–174.99	A−
155–164.99	B+
145–154.99	B
135–144.99	B−
125–134.99	C+
115–124.99	C
105–114.99	C−
95–104.99	D
85–94.99	D−
0–84.99	F

There is no curve. Your grade will depend only on how you do, and not on how well everyone else does. CS 61B is not a competition.

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.

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, return it to me (or your lab TA) with a *written note on a separate piece of paper* explaining the problem. Staple this paper to the *front* of the exam. (*Not* the back! I mean it!) The entire exam 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 must be made within two weeks after you receive the graded 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.

6.1 Lateness

We will give no credit for written homework or labs turned in after the deadline. 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 lowest two lab grades to give you leeway to handle personal crises; we don't feel obliged to offer more than that.

We do allow the projects to be turned in late, but there is a penalty. If your 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, don't 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.

6.2 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 18, for which the final exam is scheduled on Friday, May 17, from 11:30 am to 2:30 pm. The location of the final exam will be announced by the University later in the semester.

7 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, though I might not exercise it. 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. 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 or code 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. You never lose anything by giving credit generously. (Unless it's credit for writing your code for you.)

Presenting another person's work as your own constitutes cheating, whether that person is a friend, an unknown student in this or another 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. This will be punished just as harshly as stealing someone else's work.
- Working together (with someone other than your partner for the assignment) to develop a single solution and then turning in copies (or modified versions) of that solution under multiple names.
- Using Java's built-in data structure libraries, including lists, vectors, or trees, unless an assignment explicitly specifies that you may. A central goal of CS 61B is to make sure you understand how data structures work *internally*. Therefore, I expect you to use only data structures whose implementations you created yourself or learned in class. In future CS classes, you will frequently use data structure libraries whose implementations you are not familiar with; but in CS 61B, we are learning the internal mechanics of data structures.

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, you share everything with your teammates, but the rules for individuals given above apply to teams. You may not work with another team or share solutions between 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 team'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.