CS61A Summer 2009: Course Overview

Min Xu
min.cs61a@gmail.com
http://inst.eecs.berkeley.edu/∼cs61a
University of California, Berkeley
Dept. of Electrical Engineering and Computer Science

June 21, 2009

1 Introduction

The CS 61 series is an introduction to computer science. 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 softwareon 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.

Important Note: Programming ability is built by practice. Harvard Professor of mathematics Gian Carlo Rota is fond of saying "The Germans have aptly called Sitzfleisch the ability to spend endless hours at a desk doing grueling work. Sitzfleisch is considered by mathematicians to be a better guage of success than any of the attractive definitions of talent with which psychologists regale us from time to time." Computer science is no different. We expect you to spend about **20 hours a week on this class outside of lecture/lab/discussion**.

2 Course Material

The required textbook for the class is *Structure and Interpretation of Computer Programs, 2nd Edition* by Abelson and Sussman. People who do not have prior experience with Scheme may also get *Simply Scheme* by Wright and Harvey for extra help; *Simply Scheme* is entirely optional.

Important Note: The homeworks, labs, projects, solutions, lecture notes, and various other important documents will be posted on the course website at `http://inst.eecs.berkeley.edu/~cs61a`

You will receive an Instructional Computing Account (hence shortened Inst Account) to access the lab computers and servers. The lab computers runs UNIX operating system; you can find an excellent tutorial on using UNIX at `http://www.ee.surrey.ac.uk/Teaching/Unix/` . You will only need to read the Introduction and the first two tutorials.

We will use Emacs as our main text editor. You may find many detailed tutorials online but the TAs will go over the basics in the first day of lab.

Important Note: You will turn in all assignments from your inst account. Hence, it is imperative that you make sure you have an account and get familiar with it as soon as possible.

You may work at homework by remotely connecting to the lab servers by following the instructions here: (TODO:add tutorial). You can also find implementations of UCB Scheme Interpreter for your home operating system here: `http://inst.EECS.Berkeley.EDU/ scheme`

3 Lab and Discussion

The lab and discussion sections are run by Teaching Assistants; each TA will handle enrollment for his or her sections. We anticipate some rearrangements during the first week in response over/under-enrollment. If you are not enrolled, you should pick a discussion/lab section, but be prepared to shift if your first choice is full.

Please try to be in a definite 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. Please also attend the same TA's lab and discussion section for administrative sanity on our part.

4 Newsgroup and Office Hours

We will use the Google groups to provide a newsgroup for the class where you can post questions. You will receive an invite to join the newsgroup after you set up your instructional account. You do not need a Google account to access the newsgroup. The URL to the newsgroup is
`http://groups.google.com/group/cs61a-su09`

Office hours is a period of time where you can go to the GSI or the instructor to get personal help. We *highly recommend* that you go to office hours if you are having a lot of difficulties with the homework. Under the one-to-one setting of an office hour, we can narrow down the reasons that you having trouble and guide you better. Remember that we enjoy teaching so do not feel like you're intruding by going to office hours.

5 Grading

This class will be *uncurved* with the following grading scale:

The distribution of the grades are

```
Homework:  14%
Projects:  32%
```

Grade	Percentage
A+	Given at instructor's discretion
A	100% to 90%
A-	90% to 85%
B+	85% to 80%
B	80% to 72%
B-	72% to 70%
C+	70% to 68%
C	68% to 62%
C-	62% to 60%
D+	60% to 58%
D	58% to 52%
D-	52% to 50%
F	below 50%

```
2 Midterms:  16% each
Final:  22%
```

For every homework, most problems will be graded by effort with a small number graded on correctness. Each homework will be worth 2% of your final grade with 1.5% from effort-grade and 0.5% from correctness-grade. For the problems graded by effort, you must show that you have made an honest attempt at an answer; for example, if you could not get the right answer, you can write a short note indicating with which part you had difficulties. **You may very well get full effort-grade for an answer that is completely wrong so long as you show significant effort**.

The projects will be graded by correctness.

The exams will be open book and open notes. Furthermore, it will usually start in the evening and be untimed, meaning you can take as much time as you'd like. We will write the exams with the intention that most people can finish in two hours. The untimed exam rule is designed to minimize stress during exam time; please do not abuse this rule and stay until dawn or we may be forced to cancel it.

There is a separate handout that details how to submit the homeworks electronically. You may find the instruction on the course website also.

If you feel your homework or project is graded incorrectly, you may submit an online regrade request. However, you must say specifically what the mistake is. Do not just turn in a regrade request if you feel you deserve more points.

Slip Days: You have 3 slip days over the whole semester that you can use to turn in projects late **with the exception of project 4**. For example, you can use up your slip days by turning in project 1 three days late or turning in project 1, project 2, and project 3 one day late each. Turning in the group projects late will use up both yours and your partner's slip days. *You may not use slip days for homework.*

6 Policy on Collaboration

You are encouraged to work in group of two to four on the homeworks but you *must understand the problem and write up the solution yourself.* If you worked with other people, please write the name and login of everyone you collaborated with in `LOGIN.hw1` file.

Important Note: It is a violation of the department policy on academic honesty for some people in the group to just say the solution for everyone else to copy. The rule of thumb is if that someone who does not understand what you say can still write down the right answer from what you say, then you are saying too much.

You must work on the project 1 and 4 on your own. You may discuss ideas with other students but NOT the code itself. You will work on project 2 and 3 in a group of two with a partner of your choice. Again, your group may not discuss code with other groups in the class.

We will adhere to the EECS department policy on academic honesty that you can find at
`http://www.eecs.berkeley.edu/Policies/acad.dis.shtml`

7 FAQ

- **Q:** Am I qualified to take this class?
  A: You should have prior programming experience equivalent to what you would get from the most basic Computer Science course in either a high school or community college. A strong background in proof-based mathematics can substitute for programming experiences. If you have neither, you can still take the class; if you can finish the first homework then likely you will be able to handle the course.

- **Q:** Do I have to attend lecture/discussion/lab?
  A: We highly encourage you to but do not require it. We have found from previous semesters that people who do not attend lectures, labs, and discussions in general score MUCH worse on exams than people who do.

- **Q:** Do I have to turn in the lab exercises?
  A: No.

- **Q:** Can I switch to a different lab/discussion section?
  A: Only if you can get approval from that new section's TA.

- **Q:** I am not registered for the class, which section do I go to?
  A: Ones that are least crowded. Because the number of computers in lab is limited, if you are not registered for the class, we may be forced to kick you off a computer to give it to people who are.

- **Q:** Can I take the exam at a different time?
  A: Except for extenuating circumstances like medical emergency, religious holiday, conflict with another exam, etc, No.

- **Q:** I have disability and need special arrangement for the course. What should I do?
  A: Please contact the Disabled Student Program at `http://dsp.berkeley.edu/`. They will almost certainly be able to provide any accomodation you might require.

- **Q:** I hate Scheme with the fire of ten thousand suns. Why don't we use a practical language like Java?
  A: Scheme is practical; it just isn't as widely used as Java. We choose Scheme because it has very simple and elegant syntax and also because it lends itself easily to both functional and imperative programming paradigm. If you do not like Scheme; the reason is most likely that you are having trouble getting used to functional programming paradigm, not that Scheme is a bad language.

- **Q:** My project partner does nothing! Isn't it unfair that other students will influence my grade?
  A: Working in group is an important part of computer science. It is important to learn to work well with people who are not necessarily your friends. If your partner is truly atrocious, then please talk to one of the course staff.

- **Q:** Why am I having so much trouble with the homework?
  A: The homework and projects in this class are often not straightforward. They are designed to really make you think and sometimes, to make you think *creatively*. We highly encourage you to come to office hours if you thought about some problems for hours with no result.

- **Q:** What is the meaning of life?
  A: This topic is beyond the scope of this class.