CS162: Programming Languages
Winter 2017

Instructor: Kyle Dewey (kyledewey@cs.ucsb.edu)
Teaching Assistants:
- Michael Christensen (mchristensen@cs.ucsb.edu)
- Burak Kadron (kadron@cs.ucsb.edu)

Website: http://cs.ucsb.edu/~kyledewey/cs162/
Lecture: Tuesday / Thursday 5:00 PM - 6:15 PM in Girvetz Hall 2128
Discussion:
Friday 9:00 AM - 9:50 AM in Phelps 2514
Friday 10:00 AM - 10:50 AM in Girvetz Hall 1116

No Course Textbook

Course Description:
Various topics in programming languages, along with their connections to formal logic. Special emphasis is placed on the following topics:
- First-order logic
- Type systems and typechecking
- Functional programming
- Formal language semantics
- Logic programming

Prerequisites:
- CS130A
- CS138

Graded Components:
Your grade is entirely based on 7 equally weighted programming assignments. To be clear, there are no exams, no in-class assignments, and attendance is optional both for lecture and discussion. That said, you are responsible for everything covered in lecture, and the **assignments assume** you have been attending lecture. Historically, students who do not attend lecture have great difficulty with the assignments.

Assignments (subject to change):
- Introduction to Scala
- Functional programming with Functional Images
- Implementing a typechecker for a simply-typed language (`SimpleFUN`)
- Implementing a typechecker for a polymorphically-typed language (`PolyFUN`)
- Implementing a small-step interpreter for a functional language (`SimpleFUN`)
- Introduction to Prolog
- Implementing a small-step interpreter for a subset of Prolog (`miniProlog`)

Final Grade Assignment:
The table below describes how final letter grades are assigned in the course. The left column shows the minimal score necessary to receive the grade in the right column. The highest letter grade possible given the score is chosen; e.g., if you receive an 88.2, you'd receive a 'B+' for the course, which corresponds to being >= 86.5.

If your score is >=...	...you will receive...
96.5	A+
92.5	A
89.5	A-
86.5	B+
82.5	B
79.5	B-
76.5	C+
72.5	C
69.5	C-
66.5	D+
62.5	D
59.5	D-
0	F

The above cutoffs are strictly enforced; e.g., if you had a 79.4999999, this would be considered a 'C+' as opposed to a 'B-', as the cutoff for a 'B-' is 79.5. The reasons for this are twofold:

1. Ultimately, any grading system imparts a cutoff somewhere. Any relaxation introduces inconsistency, which is unfair.
2. These cutoffs are slightly lower than the typical cutoffs; e.g., the cutoff for an 'A+' is typically 97, not 96.5. As such, the cutoffs used for this class effectively have built-in rounding.

Grading Errors:
Internally, automated testing is used for a large portion of grading, so it is unlikely that there has been an error of some sort. Grading errors in this context are usually due to your code somehow not working correctly with respect to our testing environment, and will usually lead to bizarre results (e.g., all tests fail). Generally, if you pass the tests we provided without somehow modifying the mechanism used to run those tests, then you should not have these sort of problems on our end. However, on rare occasions, this happens. If you believe such an issue occurred, email us **within one week** of receiving your assignment grade. From there, we can double-check.

Regrading:
Each of the assignments allows for **regrading**, wherein a student may get back up to 1/3 of the points lost on an assignment. For example, if a 70 / 100 was originally received, a student may get this bumped up to an 80 / 100.

Regrading is applicable if an assignment was turned in and *some* attempt was made (e.g., turning in empty files or unmodified template files is not acceptable).

If you want to perform a regrade, you must submit your solution as usual with `turnin` **within one week** of receiving the grade for the assignment. In addition, you must also **email us within one week** that you are submitting a regrade.

We will record the grade for your new solution as `NewScore`. Your final score for the solution is then determined by the following equation:

```
FinalScore = OldScore + max(0, (NewScore - OldScore) / 3)
```

As shown with the above formula, if your new solution performs worse than the old solution, there is no negative adjustment to your score. That is, it is always in your best interest to submit a regrade if you did not receive full credit.

Important note about office hours and regrading: historically, office hours tend to get swamped after an assignment has been returned, due to lots of students asking questions related to regrading. In the most extreme case, even with 5x the amount of normal office hours and 2x the support, students would wait up to 45 minutes for an answer. The underlying problem was that of load balancing - most students went to a single office hour session near the regrading deadline. As such, it is recommended that if you are to submit a regrade, that you **start early**, particularly if you think you will need extra help. This will help spread resources out in a more efficient manner.

Background Behind Regrading:
Regrading exists to encourage students to revisit assignments that went poorly, in order to ensure that material is properly learned. This is a good strategy in general, but it is especially important for multiple assignments in this class which build upon each other, specifically:

- Scala is used in nearly every assignment, not just the Introduction to Scala assignment
- Components of your typechecker for `SimpleFUN` are reused in your typechecker for `PolyFUN`
- Both the `SimpleFUN` and `miniProlog` assignments share a similar structure

Regrading gives you an opportunity to improve upon an assignment and get additional feedback on it with practically no risk.

Due Dates / Late Policy:
For items turned in late, each person has 24 hours worth of "grace" time in total. For example, if someone were to submit the first assignment 4 hours late and the second assignment 6 hours late, then a total of 10 "grace" hours have been used. Both submissions would be accepted without incident, and there would be 14 "grace" hours remaining. Except in extenuating circumstances, submissions for students who have gone beyond their grace time will **not** be accepted.

A little background on this policy - the grace time is intended to be used as a sort of last-minute "oops" relating to a poor time estimate of (what should be) final touches. This policy tries to reduce the number of submissions hastily done just to meet a deadline, and to prevent issues of submissions that missed the deadline by a relatively small amount of time. It is not intended to be used as a way to extend the deadline for one assignment for a day, although it certainly can be used that way. Be forewarned: once it's gone it's gone, so use it wisely!

Extenuating Circumstances:
"Extenuating circumstances", for the purpose of this class, is defined as anything beyond our immediate control. In these cases, at my discretion I can grant an extension. To be absolutely clear, there is **no** guarantee that I will do so, and I am not obligated to grant them. For the things we can predict (e.g., trips), I expect to be contacted at least a week in advance. For the things we cannot predict (e.g., illness), I need official documentation explaining the situation (e.g., a doctor's note).

Communication Policy:
I have two office hours per week, though I may increase this to 3-4 if questions abound. I'm also available by appointment.

With email or Piazza, assume that I will take at least 24 hours to respond. Typically my response time is much, much faster than this, but I do occasionally take this long. Historically, this has only been an issue the last hours before an assignment deadline, and only for students who started far too late. The point being: start early!

Where possible and appropriate, Piazza should be preferred for communication. This allows for other students to answer questions, which usually means better response times for students.

Academic Honesty:
In as few words as possible, cheating and plagiarism will **not** be tolerated. I understand that the temptation may be high ("it's just this one assignment" or "I just need this class"), but this is **no** excuse. At the very least, this is unfair to all the students who did not resort to such unethical means, who instead took the time and struggled through. I will be following UCSB's Academic Conduct policy on this (from http://www.sa.ucsb.edu/Regulations/student_conduct.aspx, under "General Standards of Conduct"), quoted below for convenience:

> It is expected that students attending the University of California understand and subscribe to the ideal of academic integrity, and are willing to bear individual responsibility for their work. Any work (written or otherwise) submitted to fulfill an academic requirement must represent a student's original work. Any act of academic dishonesty, such as cheating or plagiarism, will subject a person to University disciplinary action. Cheating includes, but is not limited to, looking at another student's examination, referring to unauthorized notes during an exam, providing answers, having another person take an exam for you, etc. Representing the words, ideas, or concepts of another person without appropriate attribution is plagiarism. Whenever another person's written work is utilized, whether it be a single phrase or longer, quotation marks must be used and sources cited. Paraphrasing another's work, i.e., borrowing the ideas or concepts and putting them into one's "own" words, must also be acknowledged. Although a person's state of mind and intention will be considered in determining the University response to an act of academic dishonesty, this in no way lessens the responsibility of the student.

The first instance of plagiarism is an automatic zero for the assignment, with no opportunity for regrading. The second instance of plagiarism will result in an 'F' for the entire course.

On Collaboration:
All assignments are individual; there are no assignment partners. That said, it is OK to discuss **ideas** with other students, just **not code**. This means that sharing code is forbidden, as are discussions of low-level code details (e.g., define a function that takes these parameters, etc.). To be clear, on our end, it is usually impossible to tell the difference between cases where code was outright shared and cases where lots of low-level details were discussed; in both cases the end result is two very similar pieces of code.

There is occasionally a gray area here over what is acceptable to discuss or not. If you are unsure, do not hesitate to privately contact us. In an absolute worst-case scenario, if you tell us ahead of time to expect similarities between two student submissions, we will not consider it plagiarism. This means you will **not** get an automatic zero, though there may still be a grading penalty.

Seriously, Do Not Take Code, and Protect your Code from Being Taken!
We have an automated mechanism which can detect similarities between code. If the mechanism flags that two pieces of code are similar, then we will manually inspect them side-by-side. While it is possible that two codebases are similar by chance, this is rare in practice, and oftentimes it is obvious when inappropriate collusion has taken place.

On the other side of the coin, you must **protect your code from being stolen**. On CSIL, sometimes your code is publicly available, at least to anyone with access to CSIL. To guard against your code being taken, run the following command:

```
chmod 700 cs162
```

...where `cs162` contains your class code (or whatever directory you use). Additionally, if you use version control, make sure your code is not publicly accessible (as with public repositories on GitHub). On GitHub, you can get private repositories for free by filling out https://education.github.com/discount_requests/new, and Bitbucket similarly offers private repositories. Even if you obfuscate project names, filenames, or directory structure, it is still trivial to find code with a targeted search.

Historically, code has been taken both from CSIL and from public repositories multiple times. In general, we cannot figure out who took what from whom, so both parties can end up being penalized.