Course Information

This handout explains how the course is organized and administered. It describes how you will be graded and what the course staff expects of you.

1 Staff

<table>
<thead>
<tr>
<th>Position</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lecturer</td>
<td>Professor Saman Amarasinghe</td>
</tr>
<tr>
<td>Lecturer</td>
<td>Professor Charles E. Leiserson</td>
</tr>
<tr>
<td>Teaching Assistant</td>
<td>Ekanathan Palamadai Natarajan</td>
</tr>
<tr>
<td>Teaching Assistant</td>
<td>Ruben Perez</td>
</tr>
<tr>
<td>Teaching Assistant</td>
<td>Phumpong Watanaprapornkul</td>
</tr>
<tr>
<td>Lecture Scribe</td>
<td>Tao B. Schardl</td>
</tr>
<tr>
<td>Support Staff</td>
<td>Mary McDavidt</td>
</tr>
<tr>
<td>Support Staff</td>
<td>Marcia Davidson</td>
</tr>
</tbody>
</table>

2 Web

Communication: http://www.piazza.com
Email: 6172-staff@csail.mit.edu
Lecture notes: http://nb.mit.edu

We will be using the Stellar course-management system. Please make sure that you are registered with Stellar as a member of 6.172 for the Fall 2011 semester. We will use the course website to post assignments and accept written submissions.

The Piazza website provides a wiki-like service for organizing questions and answers regarding course content. The course staff will generally use Piazza for general communications with students, rather than email. Piazza allows students to submit questions to the course staff, which other students can see. Questions can be submitted with the author identified, but anonymous questions can also be submitted. The course staff can respond to questions and provide answers, but students can also respond and provide answers. You should have received an email inviting you to register on the 6.172 Piazza website. If not, please send email to the course staff. All questions about course content and administration should be posed via Piazza, not via email. If you are shy, post your question to Piazza anonymously.

We will post lecture notes on the nb website, which allows students to comment on the notes collaboratively. We will also post them on Stellar, but we encourage you to use nb to give us feedback. We are hoping to produce a good set of lecture notes this term which will form the basis of a textbook. Your feedback on the lecture notes will be greatly appreciated.

You may contact the course staff by sending email to 6172-staff@csail.mit.edu. Generally, you should not use this mailing list, however, but rather Piazza. The mailing list should be used only for personal issues.
3 Lectures

Lectures will be held on Tuesdays and Thursdays from 2:30 to 4:00 P.M. in 32-123. Please plan to attend regularly. All material covered in the lectures will be fair game for projects and quizzes. We will post lecture slides and notes on Stellar and nb, but they often will not contain all of the information presented in a particular lecture and should not be considered a substitute for attendance.

4 Recitations

We will hold regularly scheduled recitations on Fridays. The recitations are designed to be hands-on tutorials covering tools and other practical topics. Recitations are mandatory. Missing more than one without an approved excuse will result in failing the course. If you must miss a recitation, notify your TA as soon as possible. You will still need to complete the recitation assignment and have a TA check it off.

The duration of each recitation section is two hours, divided into a tutorial portion for roughly an hour and then a hands-on portion. Additional office hours will be held MWF from 4 P.M. to 6 P.M. in the G7 lounge.

5 Exams

There will be two 80-minute quizzes given during class time. The dates are on the Stellar course calendar. The quizzes will be closed book and closed notes, but you will be permitted crib sheets. Roughly speaking, each quiz will cover half of the course material. All material covered by the lectures, projects, or prerequisite courses, as well as any other material indicated by course staff, is fair game for the quizzes.

There will be no final exam.

6 Late and missing work

Since this is a fast-moving class, you will most likely find it difficult to play “catch-up” if you should fall behind. For this reason, among others, we will generally not accept late projects. You should submit whatever you have by the deadline, and we will award partial credit as appropriate.

If you find yourself in an unusual situation which you believe may constitute an extenuating circumstance, please let us know in advance. We may require confirmation from a Dean of student affairs or a medical professional.

7 Grading

Each assignment will describe how the materials you submit will be evaluated. The scores you receive on each assignment will be combined to produce your final grade after being weighted approximately as follows:
In addition, if you receive no substantial credit for the final project or for any two other assignments, you will receive a failing grade. Missing more than one recitation checkoff will also result in a failing grade.

## 8 Academic honesty

Institute guidelines relating to academic honesty require that we inform you of our expectations regarding permissible academic conduct. It is your responsibility to satisfy both the letter and the spirit of these rules. If any part of this policy is unclear, or if you have any questions or concerns, please ask a member of the course staff for clarification.

If you violate this policy, you will be referred to the Committee on Discipline to face the possibility of expulsion from MIT and other punitive actions. We take academic dishonesty extremely seriously. Please do not put us in a position where we must deal with it.

The course staff will use technological and other means to detect cheating. If one party shares material with another, we treat both the giver and receiver as equally guilty of academic dishonesty.

You may not share ideas, algorithms, approaches to solutions, or answers to written questions with anyone who is not a member of your group, whether or not they are in the class.

When working in a group, you may (of course) share ideas, code, and anything else that may be appropriate within the group, but be sure that you are making a fair contribution to your group. We ask you to briefly describe the contributions of each group member in the written material you submit with each project. If you have not made a fair contribution to that work, putting your name on your group’s work is considered academically dishonest.

You may not permit anyone besides the staff, the members of your group, and your Master (see Section 12) to view your source code, your compiled binaries, or your written documentation. You may not view anyone else’s solutions or materials. You may not copy or transcribe a solution from any source. The work you submit must be your own.

You may use general conceptual material, such as that you might obtain from a textbook, regardless of its source. For instance, *Introduction to Algorithms* by Cormen, Leiserson, Rivest, and Stein is an excellent resource for looking up algorithms. If you do use any material from an external source, please cite it briefly and clearly in your documentation.

In short, make sure that you are turning in your own work! If you have any questions or concerns, talk to the course staff. If you feel that you may have violated this policy, it will go far better for you if you report your possible transgression to us than if we find out by other means.
9 Projects

The bulk of your out-of-class time will be spent completing six projects of increasing scope and complexity. The general structure of each assignment is described below. The grading policy is described in Section 7.

Project 0 will be completed individually. It will give you an opportunity to familiarize yourself with the development environment which will save you time on future projects.

Project 1 will be completed in pairs. The project will allow you to learn improving performance using the Linux perf tool and to experiment with word-level parallelism. You will modify two small programs with the goal of improving their performance. You will investigate codeperf and efficient bitwise representation on data words.

Project 2 will be completed in pairs. You will begin by analyzing optimizing a single-threaded physical simulation. Then, you will be introduced to Cilk++, a language, compiler, library, and tool chain for developing multithreaded applications, and will parallelize your project. This project will expose you to many of the issues associated with correctness and performance in a multithreaded application.

Project 3 will be completed in pairs. This project will require you to examine the complex real-world problem of high-performance memory management. You will implement a library which provides the malloc, free, and realloc functions (that is, the C memory-management API) as efficiently as possible for a number of different plausible workloads.

Final Project will be completed in small groups of two or three students. You will work to optimize a game engine, which plays a two-player board game. Unlike in previous projects, we will provide relatively little direction. You will have to use everything you have learned during the course to identify performance issues and eliminate them. Since a higher performance program should play better, we will have a tournament to determine the winner.

10 Problem Set

There will be one take-home written assignment that will evaluate your knowledge of the theoretical material that the course will have covered. You will have a chance to demonstrate your knowledge of some of the analytical techniques and fundamental material that you will have learned.

11 Software engineering

Research on programmer productivity has demonstrated that pair programming, where two programmers sit together during coding, produces better quality code faster than two programmers coding on their own. For group projects, we encourage you to do pair programming.

Likewise, regression testing demonstrably promotes fast code development. Studies have shown that over 90% of bugs are repeat occurrences of previous bugs. The idea of regression testing is to build a battery of tests that include test cases for every bug that has ever been found in the software. Whenever you find a bug in your code, before fixing the bug, you write a test that detects the presence of the bug and add it to the regression suite. Then, you fix the bug. Finally, you run the regression tests to ensure that the new code passes the test. With a good regression suite, it’s easy to make changes to software, because if you mess something up, you don’t go far down the road before discovering the fact.

It can also be helpful to write unit tests, which test a particular function, method, or class. That way,
instead of trying to find a bug in a large codebase when it’s all harnessed together, you can divide-and-conquer to do your detective work in smaller components, which is much faster and easier.

You should also make liberal use of the `assert` package to check assumptions, including loop and recursion invariants. Liberal use of `printf` can help you see the internal state of your computation. If you use `#ifdef`’s to conditionally execute `printf`’s based on a debug flag, you will not adversely impact production running time when the debug flag is set to `FALSE`. Some programmers also find it productive to use debugging tools, such as `gdb`, which allows setting of breakpoints, inspection and modification of internal state, and single stepping.

12 MITPOSSE

The course staff has recruited senior software engineers in the region to share with you their invaluable experience and give you concrete advice on your design and code. These Masters in the Practice of Software Systems Engineering (MITPOSSE) will review your designs and code. Although they will not provide any input into your grade, their expertise can help you produce projects with a higher grade.

Your assigned Master has agreed to volunteer a significant time to help you learn. We’ve hand-picked these volunteers from among the very best software engineers in the region. They know what they are talking about. All have extensive experience with real-world, on-the-job design reviews.

Please accord these Masters your greatest respect, since they are volunteering their time. Be punctual. Be prepared. Be gracious. You can learn a lot from them.

13 Project structure and grading

Generally speaking, each assignment is arranged in the same way. First, you will complete a beta release of the assignment, implement a set of regression tests, and submit a written report along with your code and tests. Within a couple of days, the course staff will run all of the submitted code against all the tests, as well as do performance testing. You will be graded on code correctness, code cleanliness, documentation, performance, and test coverage. Over the week following your submission, you will sit down for a design and code review with one of our MITPOSSE members. Finally, about two weeks after the beta release, you will resubmit a final release of your code and documentation for a second evaluation.

Grading of the beta submission

The teaching assistants will grade your beta submission. You will get points for correctness, documentation, code cleanliness, test coverage, and performance. More detailed information will be provided with each assignment.

Your documentation should include a short but thorough “executive summary” to describe your submission. With regard to testing, if someone else’s submission passes your tests but has a bug, you will lose points. If you find a bug that no or few others find, you will receive extra points.

After examining the results of the beta, the course staff will set a performance goal for each problem. This goal will typically be set so that few of the initial submissions meet it. Any submission which has not met this performance goal will get a portion of the available points, depending on how close to the goal the submission is.
You will submit code by pushing it back into the upstream repository in the course locker (that is, by typing `git push` after committing your changes). You will submit your written reports in PDF format electronically via the course website.

**Design review**

You will arrange a meeting with your assigned Master in the week following the beta submission’s due date and provide them with a copy of your submission. When you meet with your Master (for about an hour), you will walk them through everything you did and answer any questions they have. Review your “executive summary” with them at the beginning of your meeting. Your Master is not grading you, and what they think of your efforts has no impact on your grade (although your attendance and participation does). They are only providing you with personalized advice and suggestions to help improve your grade and, of course, become a better programmer. They are also not there to write code for you or to provide exact answers or algorithmic suggestions. You should, however, feel free to ask them for critiques of your coding technique and for general advice. Although your Master’s feedback does not directly impact your grade, should you fail to attend a design review, your grade for that project will be impacted adversely.

**Grading of the final submission**

The final submission will generally be due about two weeks after the due date for the beta. To receive full credit, you must meet or exceed the performance goal announced previously and maintain (or achieve) a perfect correctness score. We will share the set of all the (correct) tests that you and your classmates produced with you for use in your regression testing.

Since this course also covers software-engineering issues related to code cleanliness, if we pointed out any non-performance-related issues with your beta (such as problems with code readability or organization), you should also correct those for the final submission. If you fail to correct these issues, you will lose points.

With the final submission, please submit a brief, informal write-up in PDF format electronically via the course website. Highlight the changes you have made so that we can be sure to examine them and give you credit.

**If you need help**

Please get started on your assignments early, since programming assignments often take longer than you expect, even if you take account of the fact that programming assignments often take longer than you expect. If you do need help, feel free to send email to the course staff. Earlier is better, since if you wait until the night before the assignment is due, you may not receive help in time for it to be useful, but first, please make a reasonable attempt to resolve the issue on your own.

---

Speed is fun! Enjoy the class!