CAS CS 210: COMPUTER SYSTEMS
SPRING 2018
Computer Science Department
College of Arts and Sciences
Boston University

Important Dates: Last Day to DROP Classes without a 'W' grade is February 22, 2018. Last Day to DROP Classes with a 'W' grade is March 30, 2018.

INSTRUCTOR Prof. Abraham Matta, 111 Cummington Mall, MCS 294A (Math & Computer Science Building). Phone: (617) 358-1062. Email: matta@bu.edu

To reach me at times other than my office hours, please send me email. I check my email regularly.

TIME AND PLACE Tuesdays and Thursdays 2-3:15pm. KCB 101. KCB is the Kenmore Classroom Building, located at 565 Commonwealth Ave.

COURSE DESCRIPTION This course takes a programmer's perspective to learn about the inner structure of computer systems, the design and implementation of abstractions that enable humans to use computers efficiently, the basics of C and assembly programming, the mapping between C and assembly, and between assembly and machine language, and the role of operating system software.

Our goal is to learn what a “beautiful” computer system is and how it works. Quoting an Italian painter named Elio Carletti: Beauty is the summation of the parts working together in such a way that nothing needed to be added, taken away or altered.

We will also learn how to become strong (“brilliant”) programmers who write fast and reliable programs. As the saying goes: Computers are incredibly fast, accurate, and stupid; humans are incredibly slow, inaccurate and brilliant; together they are powerful beyond imagination.

CS 210 is a core (group A) course for computer science majors. It provides background for courses in the systems area such as operating systems, compilers, and networks, not to mention more advanced courses in computer architecture.

PREREQUISITES This course assumes that students have a solid background in Python, Java, or C++ programming from CAS CS 111 or equivalent. CS 112 is also recommended, but not essential for students with strong programming skills. A solid working knowledge of operating systems, such as Unix/Linux and Windows, is also assumed. CS 131 or MA 293 is helpful for the material on Boolean logic and data representation, but is not essential.

We will make use of C as an example high-level language because its syntax and semantics are closer to assembly language concepts. C is more suitable for exposing low-level system details and achieving higher performance in real implementations. If you know C++, this should not distract you since C is mostly a subset of C++. C and C++ both share many of the same fundamental programming constructs. C, however, lacks support for object-oriented programming. On the other hand, if you know Java or
Python, there are aspects of C, particularly pointers, explicit dynamic memory allocation and formatted I/O, which do not exist in Java and Python. Fortunately, the C language is relatively small and there are many good references (see required text).

**SYLLABUS** This is a tentative syllabus and subject to change. Dates are approximate. Speed and level of coverage will depend to some extent on the maturity and background of the class. Unless noted otherwise, **readings indicated below are required and from the CMU textbook**; more detailed section-by-section reading assignments are noted.

<table>
<thead>
<tr>
<th>Dates</th>
<th>Topics</th>
<th>Readings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tue Thu</td>
<td><strong>Overview:</strong> Course Goals, Introduction to Computer Architecture &amp; Organization, Software Hierarchy.</td>
<td>Ch. 1</td>
</tr>
<tr>
<td>1/18</td>
<td><strong>Data Representation:</strong> Conversion between Number Systems, Character Codes, Boolean Algebra, Bit-level &amp; Logical Operations in C, Integer &amp; Floating Point Numbers. <strong>Computer Arithmetic:</strong> Addition, Subtraction, and Operations in C.</td>
<td>Ch. 2</td>
</tr>
<tr>
<td>1/23</td>
<td><strong>Instructions:</strong> Intel Instruction Set Architecture (ISA): arithmetic, data transfer, addressing modes, ...</td>
<td></td>
</tr>
<tr>
<td>1/25</td>
<td><strong>More Instructions:</strong> control instructions, procedures, Arrays, Structures, Assembly versus Machine Language. <strong>(Skip sections 3.9.2 and 3.11)</strong></td>
<td>Ch. 3</td>
</tr>
<tr>
<td>1/30</td>
<td><strong>Exam 1</strong> <strong>No class, Substitute Monday Schedule</strong></td>
<td></td>
</tr>
<tr>
<td>2/1</td>
<td><strong>Exam 2</strong> <strong>No class, Spring Recess (Sat 3/3 – Sun 3/11)</strong></td>
<td></td>
</tr>
<tr>
<td>2/6</td>
<td><strong>Memories:</strong> Memory Hierarchy, Locality, Caches: Direct Mapped &amp; Performance, Multi-level Caches. <strong>(Read sections 6.1.4, 6.2-6.5)</strong></td>
<td></td>
</tr>
<tr>
<td>2/8</td>
<td><strong>Input/Output:</strong> I/O and OS: Memory-mapped &amp; Instructions, Interrupts, Direct Memory Access (DMA), Disks, ... <strong>(Read section 6.1.2)</strong></td>
<td></td>
</tr>
<tr>
<td>2/13</td>
<td><strong>Exam 2</strong> <strong>No class, Substitute Monday Schedule</strong></td>
<td></td>
</tr>
<tr>
<td>2/22</td>
<td><strong>Exam 1</strong> <strong>No class, Substitute Monday Schedule</strong></td>
<td></td>
</tr>
<tr>
<td>2/27</td>
<td><strong>Memories:</strong> Memory Hierarchy, Locality, Caches: Direct Mapped &amp; Performance, Multi-level Caches. <strong>(Read sections 6.1.4, 6.2-6.5)</strong></td>
<td></td>
</tr>
<tr>
<td>3/1</td>
<td><strong>Input/Output:</strong> I/O and OS: Memory-mapped &amp; Instructions, Interrupts, Direct Memory Access (DMA), Disks, ... <strong>(Read section 6.1.2)</strong></td>
<td></td>
</tr>
<tr>
<td>3/7</td>
<td><strong>Exam 2</strong> <strong>No class, Substitute Monday Schedule</strong></td>
<td></td>
</tr>
<tr>
<td>3/12</td>
<td><strong>Memories:</strong> Memory Hierarchy, Locality, Caches: Direct Mapped &amp; Performance, Multi-level Caches. <strong>(Read sections 6.1.4, 6.2-6.5)</strong></td>
<td></td>
</tr>
<tr>
<td>3/19</td>
<td><strong>Input/Output:</strong> I/O and OS: Memory-mapped &amp; Instructions, Interrupts, Direct Memory Access (DMA), Disks, ... <strong>(Read section 6.1.2)</strong></td>
<td></td>
</tr>
<tr>
<td>3/26</td>
<td><strong>Exam 2</strong> <strong>No class, Substitute Monday Schedule</strong></td>
<td></td>
</tr>
<tr>
<td>4/2</td>
<td><strong>Memories:</strong> Memory Hierarchy, Locality, Caches: Direct Mapped &amp; Performance, Multi-level Caches. <strong>(Read sections 6.1.4, 6.2-6.5)</strong></td>
<td></td>
</tr>
<tr>
<td>4/9</td>
<td><strong>Input/Output:</strong> I/O and OS: Memory-mapped &amp; Instructions, Interrupts, Direct Memory Access (DMA), Disks, ... <strong>(Read section 6.1.2)</strong></td>
<td></td>
</tr>
<tr>
<td>4/16</td>
<td><strong>Exam 2</strong> <strong>No class, Substitute Monday Schedule</strong></td>
<td></td>
</tr>
<tr>
<td>4/23</td>
<td><strong>Memories:</strong> Memory Hierarchy, Locality, Caches: Direct Mapped &amp; Performance, Multi-level Caches. <strong>(Read sections 6.1.4, 6.2-6.5)</strong></td>
<td></td>
</tr>
<tr>
<td>4/30</td>
<td><strong>Exam 2</strong> <strong>No class, Substitute Monday Schedule</strong></td>
<td></td>
</tr>
<tr>
<td>5/1</td>
<td><strong>Memories:</strong> Memory Hierarchy, Locality, Caches: Direct Mapped &amp; Performance, Multi-level Caches. <strong>(Read sections 6.1.4, 6.2-6.5)</strong></td>
<td></td>
</tr>
<tr>
<td>5/7</td>
<td><strong>Exam 2</strong> <strong>No class, Substitute Monday Schedule</strong></td>
<td></td>
</tr>
<tr>
<td>5/14</td>
<td><strong>Memories:</strong> Memory Hierarchy, Locality, Caches: Direct Mapped &amp; Performance, Multi-level Caches. <strong>(Read sections 6.1.4, 6.2-6.5)</strong></td>
<td></td>
</tr>
<tr>
<td>5/21</td>
<td><strong>Exam 2</strong> <strong>No class, Substitute Monday Schedule</strong></td>
<td></td>
</tr>
<tr>
<td>5/28</td>
<td><strong>Memories:</strong> Memory Hierarchy, Locality, Caches: Direct Mapped &amp; Performance, Multi-level Caches. <strong>(Read sections 6.1.4, 6.2-6.5)</strong></td>
<td></td>
</tr>
<tr>
<td>6/4</td>
<td><strong>Exam 2</strong> <strong>No class, Substitute Monday Schedule</strong></td>
<td></td>
</tr>
<tr>
<td>6/11</td>
<td><strong>Memories:</strong> Memory Hierarchy, Locality, Caches: Direct Mapped &amp; Performance, Multi-level Caches. <strong>(Read sections 6.1.4, 6.2-6.5)</strong></td>
<td></td>
</tr>
<tr>
<td>6/18</td>
<td><strong>Exam 2</strong> <strong>No class, Substitute Monday Schedule</strong></td>
<td></td>
</tr>
<tr>
<td>6/25</td>
<td><strong>Memories:</strong> Memory Hierarchy, Locality, Caches: Direct Mapped &amp; Performance, Multi-level Caches. <strong>(Read sections 6.1.4, 6.2-6.5)</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Final Exam:** Tuesday, May 8th, 3-5pm
**OFFICE HOURS** Mondays 2:30-4pm and Wednesdays 9:30-11am. MCS 294A.

The purpose of the office hours of the Instructor and Teaching Fellows is to answer specific questions or clarify specific issues. Office hours are not to be used to fill you in on a class you skipped or to explain entire topics. Please come to class and to your discussion session.

**TEACHING FELLOWS.**  
**Katherine Zhao.**  
Email: kzhao@bu.edu  
Office Hours: Wednesdays 2:30-4pm, and Fridays 10-11:30am. EMA 302.

**Nabeel Akhtar.**  
Email: nabeel@bu.edu  
Office Hours: Tuesdays 3:30-5pm, and Thursdays 5-6:30pm. EMA 302.

The Teaching Fellows (TFs) will hold their office hours in EMA 302 (Undergraduate lab in the Engineering Annex at 730 Commonwealth Avenue on the third floor in room 302), since it is usually more convenient to answer lab-related questions.

The TFs will lead the discussion sessions. The objective is to reinforce the concepts covered in the lectures, and answer questions (or provide clarifications) regarding the homework and programming/lab assignments.

The TFs will provide, in coordination with the instructor, online resources (FAQ, helpful hints, etc.) and help sessions on certain topics.

**DISCUSSION SECTIONS** Students attend one of these sessions each week.

- Sessions in **EMA 304** (third floor of 730 Commonwealth Avenue): Mondays 8-8:50am (A7), 11:15am-12:05pm (A4), 12:20-1:10pm (A5).

Katherine will lead the discussions for A7, A3 and A4, and Nabeel for A2, A5 and A6. The following is a tentative schedule of the discussion sections. The TFs will maintain a [web page](#) with the actual schedule and related material. Unless noted otherwise, **readings indicated below are required and from the C textbook** - more detailed section-by-section reading assignments will be noted. You may be asked to submit some lab reports as part of your homework assignments or at the end of your discussion sessions.

<table>
<thead>
<tr>
<th>Dates</th>
<th>Topics</th>
<th>Readings</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/22</td>
<td>C Primer</td>
<td>Ch. 1, 2</td>
</tr>
<tr>
<td>1/29</td>
<td>GNU tools, Compilation &amp; Debugging</td>
<td>Online references</td>
</tr>
<tr>
<td>2/5</td>
<td>Types &amp; Operators + PS#1 Discussion</td>
<td>Ch. 3, 11, 13</td>
</tr>
<tr>
<td>2/12</td>
<td>Discussion of Data Lab Assignment</td>
<td></td>
</tr>
</tbody>
</table>
2/20  2/19 Holiday - Presidents’ Day  
Tuesday 2/20 is Monday’s schedule  
Exam 1 Review

2/26  Arrays, Pointers, Strings  
Ch. 6, 9, 10

3/5  **No sections - Spring Recess**

3/12  Control Flow + PS#2 Discussion  
Ch. 4, 5

3/19  Procedures, Recursion + Discussion of Assembly Lab Assignment  
Ch. 7

3/26  Structures + Discussion of Assembly Lab Assignment  
Ch. 8, 17

4/2  Exam 2 Review

4/9  Large Programs + Discussion of Performance Lab Assignment  
Ch. 12, 14

4/18  4/16 Holiday - Patriots’ Day  
Wednesday 4/18 is Monday’s schedule  
Dynamic Memory Allocation + Discussion of Performance Lab Assignment  
Ch. 16

4/23  Input / Output + PS#3 Discussion  
Ch. 15

4/30  Final Exam Review

**GRADING** Grading (except for the final exam) is done by a number of class graders, under the direct supervision of the Teaching Fellows and the Instructor. If you have an issue with a grade (homework or exam), please contact one of the Teaching Fellows. Only if the issue is not resolved to your satisfaction, please contact the Instructor. **Grades must be appealed within two weeks of receipt**, but May 1st is the last day to request a regrade.

**TUTORING HELP** You can get extra help during the TF tutoring hours scheduled in EMA room 302. Note that terminal assistants in the EMA 302 lab are not supposed to help with course material, but to maintain the lab environment.

**TEXTBOOKS**


Note: Mastering is a learning platform that requires an access code. We will use Mastering for some online homeworks, quizzes and tutorials. This package also comes with the eText. Pearson cannot guarantee that access codes purchased from other websites besides the BU bookstore, or through the Pearson site are valid.


Both books are available from the BU bookstore or from the publisher’s site.
ONLINE RESOURCES We will be using Piazza for class discussion. The system is highly catered to getting you help fast and efficiently from classmates, the TFs, and the Instructor. Rather than emailing questions to the teaching staff, you are encouraged to post your questions on Piazza. If you have any problems or feedback for the developers, email team@piazza.com. Find our class page at: https://piazza.com/bu/spring2018/cs210/home

All course material will also be accessible through Piazza. You should regularly visit Piazza for online references (including C and Unix tools) and up-to-date information regarding readings, assignments, exam-related material, announcements, etc. You can use the discussion board for asking questions and seeking clarifications, whether from the Instructor, TFs, or classmates.

For only grade reporting, we will make use of Blackboard Learn: https://learn.bu.edu You should also be able to access Pearson’s online Mastering (learning platform) directly from Blackboard using the instructions posted on Piazza.

GRADING POLICY There will be two midterm exams and one final exam, which will include all material that is covered from the beginning of the semester until the day of the exam. All exams will be closed books and closed notes, except for cheat sheet(s) that the instructor will provide or your own handwritten 8.5”x11” cheat sheet as will be announced by the instructor before each exam. There will be absolutely no make-up exams, except for medical emergencies. In that case, blue slips from Health Services will not be accepted; you must justify your medical problem with a letter from a doctor, specifying the period of time during which you were unable to attend one of the exams.

Short quizzes may be given throughout the semester to make sure you are doing the readings on time, and also as a measure of attendance. There will also be assignments and labs, which will be a combination of "pencil and paper" problems and programming assignments. There will be about 3-4 problem sets, and 3-4 programming/lab assignments in C or Intel assembly language. Problem sets may also involve smaller scale programming/lab exercises, e.g., write or debug small programs, etc.

Your final grade will be determined approximately as follows:

- 30% by the average of the midterm exams
- 20% by the final exam
- 45% by the assignments (roughly 15% on problem sets and 30% on programming)
- 5% by the quizzes/attendance

The midterm exam average will be tentatively weighted 60% of the best grade and 40% of the lower grade.

Unless automated and otherwise specified, the grading of programming assignments will be based on the following policy:

- **Program:** works correctly (50%); in-line documentation (10%); design quality (10%)
- **Design document:** how it works (5%); tradeoffs and extensions (5%)
• **Testing document**: compilation instructions (5%); thoroughness of test cases (5%)
• **Submission**: hard copy of design & testing documents (5%); e-submission (5%)

We will use csa1.bu.edu, csa2.bu.edu, and csa3.bu.edu, 64-bit Intel-compatible Linux CS machines to grade your programming/lab assignments. Although you may use your own machine, it is your responsibility to ultimately port your assignment to one of our CS machines to make sure it is graded correctly.

Each assignment will have a due date. If late submissions are allowed for an assignment, there will be **10% penalty per day for late submissions**. But, no late assignments will be accepted after one week from due date, and the last day to submit any late assignments is **May 1st**. Extensions may be granted only for religious holidays and certified medical reasons.

**No incompletes** will be given, except for reasons of dire illness shortly before the end of the course, and only if a significant amount of work has been completed (e.g., attending lectures, handing in most assignments, and attending the midterms).

**Attendance is important**: I will take attendance at any time. I will depart from the textbook and its flow on occasions, and I will not provide backup lecture notes on certain additional details that I will cover in class, so it is imperative that you attend all lectures and take careful notes.

**ACADEMIC HONESTY** Assignments must be completed **individually**. Discussion of issues, in computer systems, is encouraged, but representing the work of another person as your own is expressly forbidden. This includes "borrowing", "stealing" or "buying" programs/solutions or parts of them (whether in printed or electronic form) from others. We may use an automated plagiarism checker. Cheating will not be tolerated under any circumstances. Handing in your own work a day or two late will affect your grade far less than turning in a copy of someone else's work on time!

**Turning in solutions that are not yours**, even when you cite your (written or electronic) sources may result in a lower grade for the assignment and your action may be reported to the Academic Conduct Committee. If you are in doubt whether your action constitutes academic misconduct, please ask the instructor.

Please review Dr. Sullivan's guidelines for collaboration from CS 111 http://www.cs.bu.edu/courses/cs111/collaboration.html And see the CAS Academic Conduct Code, in particular regarding plagiarism and cheating on exams. A student suspected to violate this code will be reported to the Academic Conduct Committee, and if found culpable, the student will receive a grade of "F" for the course.

| The use of this content and related content for lecture notes, homeworks, etc. is permitted for the class CS 210 at Boston University and for non-profit educational purposes only. Any other use requires permission of the author (Abraham Matta, matta AT bu DOT edu). |

Last updated Tuesday, February 6, 2018 © 2018 Abraham Matta. All Rights Reserved.
ASSIGNMENTS

Dates are tentative, until the assignment is actually posted or handed out.

All assignments are to be completed individually. See Academic Honesty on the syllabus. When explicitly specified, you may have the option to work in teams of two on large programming assignments. Large assignments may have multiple deadlines for submission in phases.

PS=Problem Set, PA=Programming Assignment

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Out Date</th>
<th>Due Date(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intro to Mastering</td>
<td>Wed 1/24</td>
<td>Mon 1/29</td>
</tr>
<tr>
<td>PA#0 (First C program)</td>
<td>Fri 1/26</td>
<td>Fri 2/2</td>
</tr>
<tr>
<td>PS#1 (Representation &amp; Arithmetic)</td>
<td>Fri 2/2</td>
<td>Fri 2/9</td>
</tr>
<tr>
<td>PA#1 (Data Lab)</td>
<td>Fri 2/9</td>
<td>Fri 2/16</td>
</tr>
</tbody>
</table>

EXAM 1 on Thursday 2/22

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Date</th>
<th>Date(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS#2 (Assembly)</td>
<td>Fri 3/2</td>
<td>Wed 3/21</td>
</tr>
<tr>
<td>PA#2 (Assembly Lab)</td>
<td>Wed 3/21</td>
<td>Fri 3/30 &amp; Sun 4/8</td>
</tr>
</tbody>
</table>

EXAM 2 on Tuesday 4/3

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Date</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA#3 (Performance Lab)</td>
<td>Fri 4/6</td>
<td>Fri 4/20</td>
</tr>
<tr>
<td>PS#3 (Memory)</td>
<td>Fri 4/20</td>
<td>Fri 4/27</td>
</tr>
</tbody>
</table>

FINAL EXAM on Tuesday 5/8

GUIDELINES ON SUBMISSIONS OF ASSIGNMENTS

Assignments will be available on-line.

Written homework assignments and hard copies of your programming/lab documentation / reports are to be handed in using the slotted homework (drop-off) inbox (located at the bottom), labeled "CS-210", in the hall outside of MCS 137 by 1:00 pm on the day they are due (unless otherwise specified). Late assignments must be time-stamped by a CS Office Staff and left in one of the Teaching Fellows’ mailbox (neither in the CS-210 slotted homework inbox nor the Instructor’s mailbox). Do not hand in your assignment in the class or during office hours. Do not hand in your assignment by slipping it under the office door of the Instructor.

Parts of the homeworks may be assigned and submitted online using Pearson’s Mastering.

Graded assignments will be available for pick-up during your discussion lab or during the office hours of one of the TFs.

If you believe that there is a chance that your assignment will be lost by the course staff, then here are two ways you can protect yourself: (1) make a copy of your assignment
before handing it in, and have the CS office time-stamp your copy of the assignment; or (2) “gsubmit” an electronic version of your submission (see “gsubmit” notes below) – you can type and submit your write-up in text or PDF, or if handwritten, you can scan your write-up and submit a PDF. The timestamp on your electronic submission should also indicate submission by the due date/time. Claims for “lost” assignments will be considered only if accompanied by a time-stamped (hard or electronic) copy of what you handed in. There are no exceptions to this rule.

Check for Assignments regularly. Start early!

PROGRAMMING / LAB ASSIGNMENTS

You will be required to submit an electronic copy of your code, in addition to both electronic and hard copies of supporting documentation and any requested written report(s).

When explicitly specified, you may have the option to work in teams of two on large programming assignments.

General Requirements on What to Submit

Unless automated and otherwise specified, the program you submit should work correctly and be documented. You should submit an electronic copy of the following:

- **Program**: a program listing containing in-line documentation (i.e., comments).

- **Design document**: a separate file (a page or so) describing the overall program design, a verbal description of “how it works”, and design tradeoffs considered and made. Also, describe possible improvements and extensions to your program (and sketch how they might be made).

- **Testing document**: a separate file describing how to run your program. Specify the steps that must be followed to successfully run your program. Also, describe the tests you ran on your program to convince yourself that it is indeed correct. Also, describe any cases for which your program is known not to work correctly.

It is fine to submit all the above documentation in one README file, given you have clear subtitles. To save trees, you are required to submit a hard copy only of the above supporting documentation and any requested written report(s), but not of your program listing/code.

HOW TO SUBMIT AN ELECTRONIC COPY *(Only plain ASCII files!)*

[Adapted from D. Metcalf’s note]

To submit your assignments, use the **gsubmit** program from your csa account.
gsubmit is an electronic file submission engine which will submit files or directories of files to the grader so they can be marked.

Every file submitted by a given student for a given assignment should have a unique file name. If a file is submitted with a duplicate name it will either overwrite the file or generate an error message.

To make it easy for the grader to find the files relating to a specific assignment, all files for each assignment should be stored in a subdirectory called ps1, ps2, pa1, pa2, pa3, etc. and the entire directory should be submitted.

**To submit an assignment:**
- Create a subdirectory "pa#", where # is the assignment number. This is done using the mkdir command: e.g., `mkdir pa5`
- Copy all files necessary for that assignment into the new subdirectory, using the cp command: e.g., `cp prog1.c pa5`
- Be sure to copy only the files you need to submit into this subdirectory.
- Use gsubmit to submit the entire subdirectory: `gsubmit cs210 -cp pa5`
  If submission is successful a status message will be printed.

**To submit a file to an already-submitted subdirectory:**
If you only submitted part of the assignment and would like to add another file:
- To submit a file README.txt to subdirectory pa5, type (at the prompt): `gsubmit cs210 -cp README.txt pa5`

**To resubmit a file:**
- To resubmit a file prog1.c in subdirectory pa5, first un-submit the file: `gsubmit cs210 -rm pa5/prog1.c`
- Then resubmit it: `gsubmit cs210 -cp prog1.c pa5`

**List all files which you have submitted:**
- To list all files that you have submitted, type: `gsubmit cs210 -ls`

**Looking at a file which has already been submitted:**
- To look at a file that has already been submitted, type: `gsubmit cs210 -cat pa5/prog1.c`
- You can store this in a file foo.c by typing `gsubmit cs210 -cat pa5/prog1.c > foo.c`

**Where do submitted files go?**
Each student who submits an assignment has a subdirectory created to hold his/her files, in a directory for the specified course. This is called the student's “submission spool directory”.

**How can the grader tell when a file has been submitted?**
Every gsubmit command is automatically logged in a log file, along with a time stamp.

**For further information:**
Note - The information in this document is taken from the gsubmit man page.
For further information, type `man gsubmit`.