15-213 / 15-513 / 14-513:
Introduction to Computer Systems

15-213 / 15-513 Prof. Phillip Gibbons & Prof. Seth Goldstein
14-513 Prof. Taha Khan

Carnegie Mellon University
Fall 2026

1 Organization

Class web page: https://www.cs.cmu.edu/~213/
Canvas: https://www.cs.cmu.edu/~213/external/canvas
Ed: https://www.cs.cmu.edu/~213/external/piazza
Autolab: https://autolab.andrew.cmu.edu/courses/15213-f26
Office Hours: Please see the class web page for instructor and TA office hours.
Recitations: Please see the class web page for the list of recitations.
Lectures:

- 15-213 and 15-213 students: Tue and Thur, 12:30pm–1:50pm Eastern Time, GHC 4401
- 14-513 students: Tue and Thur, 12:30pm–1:50pm Eastern Time, CIC 1201

2 Objectives

Our aim in the course is to help you become a better programmer by teaching you the basic concepts underlying all computer systems. We want you to learn what really happens when your programs run, so that when things go wrong (as they always do) you will have the intellectual tools to solve the problem.

Why do you need to understand computer systems if you do all of your programming in high level languages? In most of computer science, we're pushed to make abstractions and stay within their frameworks. But, any abstraction ignores effects that can become critical. As an analogy, Newtonian mechanics ignores relativistic effects. The Newtonian abstraction is completely appropriate for bodies moving at less than $0.1c$, but higher speeds require working at a greater level of detail.

The following "realities" are some of the major areas where the abstractions you've learned in previous classes break down:

1. *Ints are not integers, Floats are not reals.* Our finite representations of numbers have significant limitations, and because of these limitations we sometimes have to think in terms of bit-level representations.
2. *You've got to know assembly language.* Even if you never write programs in assembly, The behavior of a program cannot be understood sometimes purely based on the abstraction of a high-level language. Further, understanding the effects of bugs requires familiarity with the machine-level model.
3. *Memory matters.* Computer memory is not unbounded. It must be allocated and managed. Memory referencing errors are especially pernicious. An erroneous updating of one object can cause a change

in some logically unrelated object. Also, the combination of caching and virtual memory provides the functionality of a uniform unbounded address space, but not the performance.

4. *There is more to performance than asymptotic complexity.* Constant factors also matter. There are systematic ways to evaluate and improve program performance.

5. *Computers do more than execute instructions.* They also need to get data in and out and they interact with other systems over networks.

By the end of the course, you will understand these "realities" in some detail. As a result, you will be prepared to take any of the upper-level systems classes at Carnegie Mellon (both CS and ECE). Even more important, you will have learned skills and knowledge that will help you throughout your career.

In detail, we set forth the following learning objectives, as activities you should be able to do after completing the course:

1. Explain common bit-level representations of numeric values (unsigned, two's complement, floating point) and the consequent mathematical properties of arithmetic and bit-level operations on them.

2. Recognize the relation between programs expressed in C and in assembly code, including the implementation of expressions, control, procedures, and data structures.

3. Demonstrate ability to understand basic intention of a program through its binary representation and apply these skills to debugging programs.

4. Investigate the programmer's interaction with the underlying system through the different APIs and abstractions, including system support for process and thread control, virtual memory, and networking.

5. Analyze the consequences of imperfect system usage, such as poor memory and CPU performance, crashes, and security vulnerabilities.

6. Apply tools, both standard and self-developed, that will aid program development, including compilers, code analyzers, debuggers, consistency checkers, and profilers.

7. Apply these analytic and tool-use abilities to create reliable and efficient programs exercising the different components of a modern computing system.

8. Understand the sources of conflict that can arise when multiple threads of execution share resources, and demonstrate the ability to use synchronization constructs to mediate those conflicts.

3 Textbook

The primary textbook for the course is

> Randal E. Bryant and David R. O'Hallaron, *Computer Systems: A Programmer's Perspective, Third Edition (CS:APP3e)*, Pearson, 2016 (ISBN 0-13-409266-X).

A new paper copy of the textbook currently costs $186.66 if purchased directly from Pearson. (Note: the supplemental "MasteringEngineering" product is **not needed** for this course.) An electronic copy is

available for 180-day "rental" for $64.99, or permanent purchase for $94.99, from VitalSource. The CMU Libraries have electronic copies on reserve for 15-213/15-513/14-513 students. Their version is a scan of the paper book, so it may be harder to read than VitalSource's version, but on the other hand, it's free.

Used copies of the book may be available from various sources for cheaper than the above prices, but please make sure to get a copy of the third edition. The first and second editions (published 2003 and 2011) are quite different, particularly in their discussion of assembly language programming, and may confuse you.

Please also make sure you are reading the **hardcover** version of the third edition (ISBN 0-13-409266-X). The paperback version (also sometimes known as the international version) contains a completely different set of practice and homework problems, many of which are badly posed or even incorrect. Amazon's "Kindle edition" of the book was based on the paperback version and has the same problem.

Bryant and O'Halloran maintain a website for CS:APP containing errata, additional "Web asides," and additional reference material for students.

In addition, we expect you to use reference material about the C programming language. Our suggested reference is:

> Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language, Second Edition*, Prentice Hall, 1988.

This is the classic *K & R* book, the standard against which all reference manuals are compared. This book should be in the library of anyone who programs in C. Copies are on reserve in the Sorrells Engineering Library.

4 Course Organization

Your participation in the course will involve these forms of activity:

1. Attending the lectures and recitations
2. Doing laboratory assignments
3. Doing written assignments and peer reviewing written assignments
4. Reading the text
5. Taking a midterm exam and a final exam

Attendance will not be taken at the lectures. You will be considered responsible for all material presented at the lectures.

Lectures will typically cover higher-level concepts, though there may also be times that are applied, covering important "how-to's", especially in using tools that will help you do the labs.

The textbook contains both *practice problems* within the chapter text and *homework problems* at the end of each chapter. The intention is that you work on the practice problems as you are reading the book. The answers to these problems are at the end of each chapter. Our experience has been that trying out the concepts on simple examples helps make the ideas more concrete. Try out the practice problems associated with the readings for each class and ask questions about them at the next class.

The only graded assignments in this class will be a set of eight labs and a set of ten written assignments. Some of the labs are fairly short, requiring just one week, while others are more ambitious, requiring several weeks. The written assignments are 1-3 questions each. There are also 10 in-class paper quizzes, a midterm exam, and a final exam.

5 Getting Help

We will use the class website (https://www.cs.cmu.edu/~213/) as the central repository for all information about the class.

For technical (lectures, assignments, exam) or logistics (accounts) questions, post a question on ed. By default, any question you post will be private to you and the instructors. We will put posts on ed and in the FAQ web page answering some common questions. Be sure to check these before contacting an instructor.

The labs are offered through a hosted autograding service, developed by Dave O'Hallaron and a group of CMU undergrads, called *Autolab*. See the Autolab web page at https://autolabproject.com/.

If you want to talk to a staff member in person, the posted office hours are the best opportunity, as they represent times when we guarantee that we will be available to meet, either in the location identified or via Zoom. If a meeting is needed outside of the office hours, please use email to arrange a time.

6 Policies

6.1 Working Alone on Assignments

You will work on all assignments by yourself.

6.2 Version Control

We will be using GitHub Education for you to work on labs, with pre-populated directories for labs 4–8. The GIT repositories are private and will be deleted after the end of the semester. You will have a chance to download their contents before they will be deleted. We will explain the proper usage of the server and help with setting up the server via ed posts, in office hours and during the recitations. In general, you should work as follows:

- Add all of the provided *source* files in your lab assignment upon downloading them from Autolab and commit the initial version. Source files include any code (extensions '.c', '.h', '.pl', '.py', and '.sh'), as well as any Makefile and any program input file. It does not include any compiled libraries or reference programs.
- Commit early and often. Make it a habit to commit at least every hour you work actively on the assignment, and commit in small increments. Commit at the end of your work day.
- Make sure you commit your final version right before you submit via Autolab.

It is good software engineering practice to use version control, and learning it before starting Lab 1 is a a good idea. We may review commit statistics on the server and reach out to students who disregard our version control policy.

6.3 Handing in Assignments

All assignments are due at 11:59pm Eastern Time on the specified due date. All programming handins are electronic using the Autolab system. Note that Autolab may convert the deadline into your local time, you are still responsible for submitting before the deadline. You may hand in labs as often you like, with your most recent handin counting for credit.

Written Assignments

Written assignments will be submitted using Canvas, and are due by the due date. There are no late submissions and Canvas will generally just accept one submission. Please be aware of these restrictions when working on your submission and your peer reviews.

Late Programming Assignments

The penalty for late programming assignments (labs) is 15% per day. Each student will receive a budget of five *grace days* for the course. These grace days are provided to allow you to cope with most emergencies that prevent completing a lab on time, including computer problems, a cold, getting stuck at the airport, etc. Here is how grace days work:

- Each lab assignment has a maximum number of grace days that can be applied, ranging from 0 to 2. The grace day limits are indicated on the Labs web page and in the lab writeups.
- Grace days are applied automatically until you run out.
- If your last handin is one day late, and you have at least one remaining grace day, then you will receive full credit for the lab and automatically spend one grace day. For example, if the lab is due at 11:59pm ET on Thursday and your last handin is noon ET on Friday, then you will receive full credit and spend one grace day.
- Once you have spent your grace days, or exhausted the limit for the assignment in question, then you will receive a penalty of 15% for each subsequent late day. For example, suppose you have only one grace day left. If a lab is due at 11:59pm ET on Thursday and your last handin is noon ET on Saturday, then you will spend your one remaining grace day and be penalized 15%. If your last handin is noon ET on Sunday, then you will spend one grace day and be penalized 30%.
- Handins will not be accepted after the *end date* of the lab, which is typically three days after the due date.

Grace days are a tool to allow you to manage your time in the face of personal issues and to help smooth out burstiness in assignment due dates across classes. They are for when you are sick, when a short-term emergency situation arises, when you have too many deadlines all at once, etc. Except for serious persistent personal issues (see below), you should not anticipate additional deadline leniency. **We strongly recommend that you conserve your grace days, saving them for the more difficult assignments at the end of the term.** Grace days and penalties are computed automatically by Autolab, with grace days being used up before penalties are applied. You cannot choose how to distribute your grace days among your labs.

6.4 Dealing with Serious Persistent Personal Issues

We hope that everyone in the course will remain happy and healthy. But, if you have a serious persistent personal issue, such as being hospitalized for an extended period or needing to leave the country for a family matter, please talk to your academics advisor as soon as possible. Such issues consistently affect one's ability to succeed in all classes, rather than just the course, and the academic advisors are equipped to coordinate plans for dealing with them. We will cooperate with such plans, but we cannot construct them independently of the academic advisors. Please contact your course instructor if you are unable to keep up with the course due to a serious personal issue. A form will be provided that is checked regularly for extension requests, and extensions are granted in almost all cases when students follow these policies.

6.5 Requesting a Regrade for an Assignment or an Exam

After each lab, written assignment, or exam is graded, your score will be posted on the Autolab gradebook, Canvas, or Gradescope respectively. We will make the utmost effort to be fair and consistent in our grading. But, we are human. If you believe that you did not receive appropriate credit for an assignment or an exam, you may request a regrade as follows:

- **Lab or Written Assignment regrade request:** Post a regrade request as a private message on Ed. Provide a detailed explanation of why you believe your grade did not conform to the posted grading standard, this includes cases where your final submission autograded worse than expected, so always verify your submissions.
- **Exam regrade request:** All exam regrades must be completed using the exam system (typically, Gradescope).
- **Verbal and email requests will NOT be accepted.**
- All regrade requests must be received within **seven days** of the grades becoming available.

Your request will be processed off-line, and we will respond to your request as quickly as possible (typically within a week). This regrade policy is designed to correct legitimate mistakes in grading, while discouraging frivolous regrade requests (for the sake of being fair and consistent across the entire class).

6.6 Final Grade Assignment

Each student will receive a numeric score for the course, based on a weighted average of the following:

- **Labs (50%):** There are a total of eight labs, which will count a combined total of 50% of your score. Labs have different weightings, based on our perception of the relative effort required. See the Labs web page for the lab weightings.
- **Written Assignments (6%):** There are a total of ten written assignments, which will be graded by peer review. Your grade on an individual written assignment will be determined by three equally-weighted parts: (i) completed assignment demonstrates some effort, (ii) max score by peer reviewers, and (iii) completion of peer reviews with constructive feedback. We will drop your two lowest written assignment scores.

- **In-Class Paper Quizzes (4%):** There will be ten in-class paper quizzes, and your lowest two quizzes will be dropped. Each quiz is equally weighted, worth 0.5% of your grade.
- **Exams (40%):** There will be two exams in total, a midterm and a final. The midterm exam will be an in-class paper exam, worth 15% of the overall grade. The final exam will be cumulative and will take place in-person during finals week. The final exam will account for 25% of the overall grade.

The grading cutoff points are: 90 (A), 80 (B), 70 (C), 60 (D).

In-class (Canvas) quizzes: In the middle of many of the lectures with no in-class paper quiz, we will pause a few minutes to have students answer a 2–3 multiple-choice question quiz on the material covered in class. This will help the instructor gauge which topics need further discussion. Quiz questions will be answered by each student using Canvas. Student grades on these quizzes will be used solely as "bonus points," as discussed next.

Bonus points: We will selectively consider raising individual grades for students just below the cutoffs based on factors such as attendance (primary), class participation, improvement throughout the course, final exam performance, and special circumstances. In particular, the in-class (Canvas) quizzes provide a record of lecture attendance and attention, and hence are a good means for a student just below a cutoff to have his/her grade raised.

6.7 Academic Integrity

Please read this carefully, especially if this is your first semester at CMU!

This course, as one of a set of related systems courses in CS, ECE, and INI have adopted a uniform policy on Academic Integrity Violations (AIVs). It is available at:

https://www.cs.cmu.edu/~213/academicintegrity.html

It provides very specific guidelines on what forms of collaborations are permitted, and what forms are not. Continue referring to it during the semester as you encounter specific choices you must make in doing your assignments. The following text is based on this web document, but it also includes more specific information about the possible penalties when an infraction occurs.

Our policy is based on the following beliefs, gained through many years of experience:

- Understanding the operation and implementation of computer systems is best learned by hands-on activities: writing, debugging, measuring, and exercising programs that expose the relevant system principles.
- Developing programs from scratch, or with limited starting code, requires using design principles and logical thinking that are much deeper than can be gained by copying and modifying an existing implementation. Making use of unauthorized sources diminishes the educational value of an assignment.
- Although teamwork and collaboration are important real-world skills, it is important to first gain the core competencies that enable individuals to serve as effective team members. These courses are designed to teach and assess these core competencies. Unauthorized collaboration diminishes the educational experience and the reliability of our assessments.

Based on these principles, we provide the following guidelines on what forms of resource use, resource sharing, and collaboration are permitted in these course.

Exams

Each exam must be the sole work of the student taking it. No collaboration of any form is allowed on an exam. Students may not discuss any aspect of any exam question with someone who has not yet taken the exam.

Labs and other Assignments: Information Sources

As a general principle, you may not obtain any information about an assignment from an unauthorized source. The following provide clarifications as to which sources are authorized, and which are not. These rules hold throughout the term.

Copying:

- You **may** use material that we explicitly provide you for the assignment. No attribution is required.
- You **may** use other course material, including lectures, Piazza posts by the instructors, and material from the course website. For any such use involving code, you must provide clear attribution, indicating the source, and where the included material begins and ends.
- You **may** use any material from the CS:APP book, any other course textbook, or the CS:APP web site. For any such use involving code, you must provide clear attribution, indicating the source, and where the included material begins and ends.
- You **may not** obtain code or other solution information from an unauthorized external source, including web pages, code repositories, blog posts, AI tools, etc.

Searching:

- You **may** search for or refer to general information, including the use of systems, networks, compilers, debuggers, profilers, and program libraries.
- You **may not** search the Web for solutions or for any advice on how to solve an assignment, including from AI tools.

Reusing:

- You **may** reuse elements of general knowledge from prior courses. For example, you may use existing code for a linked list or to process commandline arguments. For any such use involving code, you must provide clear attribution, indicating the source, and where the included material begins and ends.
- If you have worked on one of the labs from CS:APP, either at CMU, at some other school, or on your own, you should arrange a meeting with one of the instructors at the beginning of the term to devise a policy on which parts of your solutions you may use. Reuse without explicit permission of an instructor, even if it's your own code, is forbidden.

Using other's code or documents:

- You **may not** look at someone else's code (or other documents.) This includes one person looking at code and describing it to another. There is no notion of looking "too much," since no looking is allowed at all.
- You **may not** make use of any information about the labs posted online, except for the authorized sources listed above.

Assistance:

- You **may** get assistance on an assignment from the instructors, course staff, and university tutors.
- You **may only** get high-level, strategic advice from others, including current and former students, and people external to the university. Forbidden forms of advice include: anything more detailed than a brief verbal description or block diagram, any kind of code or pseudo-code, explicit directions on how to assemble allowed blocks of code, and code-level debugging assistance.

Labs and assignments: Sharing and Collaborating

As a general principle, you **may not** provide detailed help with an assignment to students this semester or in future semesters for any of the above-listed courses (unless you are serving as a teaching assistant or instructor for the course.) The following are clarifications about which forms of aid are authorized and which are not. Note that they apply to *all* of the above listed courses and *from now and into the indefinite future*.

Sharing:

- You **may not** supply a copy of a file or document to an individual student or via a public channel, such as a blog post.

Providing Access:

- You **may not** have any of your solution files in unprotected directories or in unprotected code repositories, either by putting files in an unprotected location or by allowing protections to lapse. Be sure to store your work in protected directories, and log off when you leave an open cluster, to prevent others from copying your work. If you make use of a code repository, such as Github, make sure your work is kept private, even after you have left CMU.

Coaching, Assisting, and Collaborating:

- You **may not** provide electronic, verbal, or written descriptions of code or other solution information.
- You **may** clarify ambiguities or vague points in class handouts or textbooks.
- You **may** help others use the computer systems, networks, compilers, debuggers, profilers, code libraries, and other system facilities.
- You **may** discuss and provide general, strategic advice about an assignment. Providing anything more detailed than a brief description or a block diagram is not allowed. Providing any kind of code or pseudo-code is not allowed. Providing explicit directions on how to assemble allowed blocks of code is forbidden.

- You **may** provide suggestions of potential bugs based on high-level symptoms. Code-based debugging assistance is forbidden.

AI Tools

AI Tools are a valuable tool for supporting both novices and experts. However, their usage can easily lead students into violating the above rules, especially as those tools may be trained using sources that would be disallowed to directly be used. If you do use an AI Tool, you should confine your interaction along the same rules as other external sources, such as only having high-level discussions and making sure to cite and document your usage.

In brief, students should avoid any copy-paste interactions with AI Tools, and cite what they do.

You **may not** upload the code you write for a lab assignment (or even parts of it) to any AI tool (cloud-based or locally hosted or in your IDE) nor give the AI tool access to the code (or even parts of it). Much like asking a friend for help and providing access to your code, the help you get is most likely going to be an AIV, so it is not permitted. Examples of AI Tools include ChatGPT, github copilot, Gemini, Claude, and many others.

Enforcement

We will aggressively employ cheat checkers and other means to detect unauthorized use of code from this term, previous terms, and available online. All infractions will lead to formal reporting of an AIV to the university and to the program. The standard penalty will be to be given a failing grade for the course. Lesser penalties may occur, depending on the circumstances, but as a general principle, **the penalty will always be worse than if you had not turned in the assignment at all.**

The above stated rules apply even after you have completed the course. You may not share code you have written for this course with future students. That means you cannot leave your code in unprotected repositories or post it on any web page. You may not provide coaching to future students. The university policies on academic integrity include the possibility of receiving an AIV even after a student has completed a course, potentially changing a grade retroactively and even revoking a degree. We can and will pursue AIVs against students after they have completed the course, possibly changing their grades.

In risking an AIV, you jeopardize your participation in this course, your time at CMU, and your career beyond. The temptation to cheat can be very strong when deadlines approach, and you are unable to make satisfactory progress. Doing so is far worse than failing to complete the assignment.

"Regrets" Policy

Because we can all make bad decisions under the crunch of a deadline, we offer a "regrets" policy, which works as follows.

A student **may** email a course instructor with a request to withdraw a submitted assignment, even after the assignment deadline. The email request must be received by the instructor before any instructor or TA has discovered a potential AIV for the assignment. The submitted work will be deleted from autolab/canvas (no questions asked), the student will receive a 0 on the assignment, and no AIV will be filed.

7 Mobile devices and other distractions

Research on learning shows that unexpected noises and movement automatically divert and capture people's attention, which means you are affecting everyone's learning experience if your cell phone, pager, laptop, etc. makes noise or is visually distracting during class. For this reason, we allow you to take notes on your laptop, but insist that you turn the sound off so that you do not disrupt other students' learning. If you are doing anything other than taking notes on your laptop, please sit in the back row so that other students are not distracted by your screen.

8 No recording of class meetings

Recordings of any lecture or recitation, in part or whole, including any audio and/or video recordings, regardless of the media or format, and regardless of the intended or actual use, are not permitted without explicit prior written consent of all instructors. The class will be notified in advance should any such recording be approved. Students have no right to record classes under any University policy. If a student believes that he/she is disabled and needs to record or tape classroom activities, he/she should contact the Office of Equal Opportunity Services, Disability Resources to request an appropriate accommodation.

The penalty for violating this policy is an R in the course. If you are not comfortable with this, please drop the course now.

This policy is intended primarily to protect the privacy of the students. For example, no student should run the risk of potential employers finding a question, incorrect answer, or even look of confusion on the Web. The classroom is a learning environment, not an exhibition. Rather than attempt to control the uncontrollable or distinguish between neutral and detrimental uses, all recording is prohibited. Experience has shown that, excluding special cases such as use by students with disabilities or distance learners, undergraduate students do not improve their performance through the use of recordings.

9 Facilities: Intel Computer Systems Cluster

Intel Corp. has generously donated a cluster of Linux-based 64-bit multicore Nehalem servers, specifically for this class, that we will use for all labs and assignments. The class web page has details.

10 Class Schedule

Please see the schedule maintained on the class web page for information about lectures, reading assignments, suggested homework problems, lab start and end dates, and the lecturer for each class. The reading assignments are all from the CS:APP3e book.

11 Educational Research

For these courses (15-213, 15-513, 14-513), the Eberly Center is working with professors Taha Khan (14-513 instructor), Seth Goldstein (15-213/15-513 instructor) and Brian Railing (long-time 15-213/15-513 instructor) on educational research. This research will involve your coursework. You will not be asked to do

anything above and beyond the normal learning activities and assignments that are part of this course. You are free not to participate in this research, and your participation will have no influence on your grade for this course or your academic career at CMU. If you do not wish your course work to be used as research data or if you are under 18 years of age, please send an email to Chad Hershock (hershock@andrew.cmu.edu), and then your data will not be included. You will still be required to complete any activities assigned by the instructor as part of the learning experience, but your data will not be analyzed for research purposes. Participants will not receive any compensation. The data collected as part of this research will include student grades. Data analysis of coursework will be conducted after the course is over and final grades are submitted. In the future, once we have removed all identifiable information from your data, we may use the data for our future research studies, or we may distribute the data to other researchers for their research studies. The Eberly Center may provide support on this research project regarding data analysis and interpretation. The Eberly Center for Teaching Excellence & Educational Innovation is located on the CMU-Pittsburgh Campus and its mission is to support the professional development of all CMU instructors regarding teaching and learning. To minimize the risk of breach of confidentiality, the Eberly Center will never have access to data from this course containing your personal identifiers. All data will be analyzed in de-identified form and presented in the aggregate, without any personal identifiers. If you have questions pertaining to your rights as a research participant, or to report concerns to this study, please contact Chad Hershock (hershock@andrew.cmu.edu).

Plain language interpretation:

- After the semester is over, the data generated by students in this course, which often comes from things like assignments, projects, surveys, etc., is stripped of all identifying information and aggregated into a larger research dataset for analysis.

- As a potential participant in this research, you have a say in what happens to your data. If you are OK with data generated by you in this course being de-identified and aggregated into a larger research dataset, then there is nothing you need to do - simply proceed through the course as you would with any other course.

- IF you would NOT like your data to be used for that purpose (or you are under 18) that is when you email Chad Hershock and say "Hi Chad, this is my name and course number, I would like to opt-out, thanks, goodbye", and your data will not be included in the research analyses.

- This opt-out process is confidential with the Eberly Center, and your instructor(s) will not know whether you have opted out of the research study or not.

- Importantly, your decision in this matter will NOT affect your experience in the course. You are only making a decision about what happens to your data AFTER the course is over.

Below are some potential questions students may have...

Question	Answer
What do I need to do?	If you would like to opt out of your data being used in research analyses, all you need to do is email Chad Hershock (email is in the syllabus) with your **name, course number, and say "I'd like to opt out". If you do not wish to opt out, you do not need to do anything.**
What is this research about?	This research is focused on better understanding teaching and learning at CMU. The data will be used for various projects on educational interventions and measured impacts on learning and/or attitudes. The specifics of this educational research may vary from course to course or instructor to instructor. If, after the course is over, you're curious about this kind of work, please contact Chad Hershock or the Eberly Center (eberly-assist@andrew.cmu.edu) who can provide additional details.
Do I have to make up my mind right now?	No, there is no need to make up your mind right now. You can choose to opt out anytime, even if it is on the last day of the semester.
What if I don't want to participate, what happens?	If you'd like to opt out, the only thing that will change is what happens to your data AFTER the course is over. **Your required coursework will be the same regardless of your decision.**
Can I see the results?	Often the results from this kind of work do not come together right away. If you are curious about the results after this course is over, please feel free to contact your course instructor or the Eberly Center and we would be happy to give you an update, if possible.
How will the data be used?	In two ways: to help improve the course and to contribute to educational research on teaching and learning. Note that all analyses occur after course grades are submitted and student identifiers are removed.
If I opt out, do I still have to complete work assigned by the instructor?	Yes.