CSE 250: Course Overview, Logistics

Lecture 0

Guest Lecturer: **Aaron Huber**

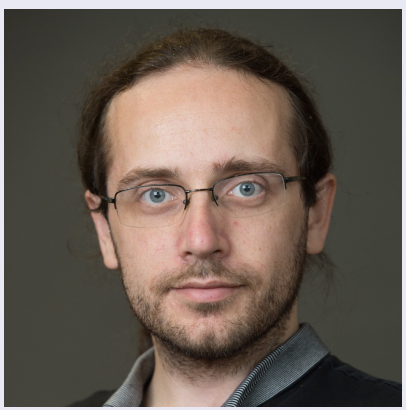
Aug 28, 2023

Who are we?

Hi, I'm Aaron.

- Online Data Interactions Lab [https://odin.cse.buffalo.edu]
  Current Research Interest: Incomplete/probabilistic databases

This is Oliver



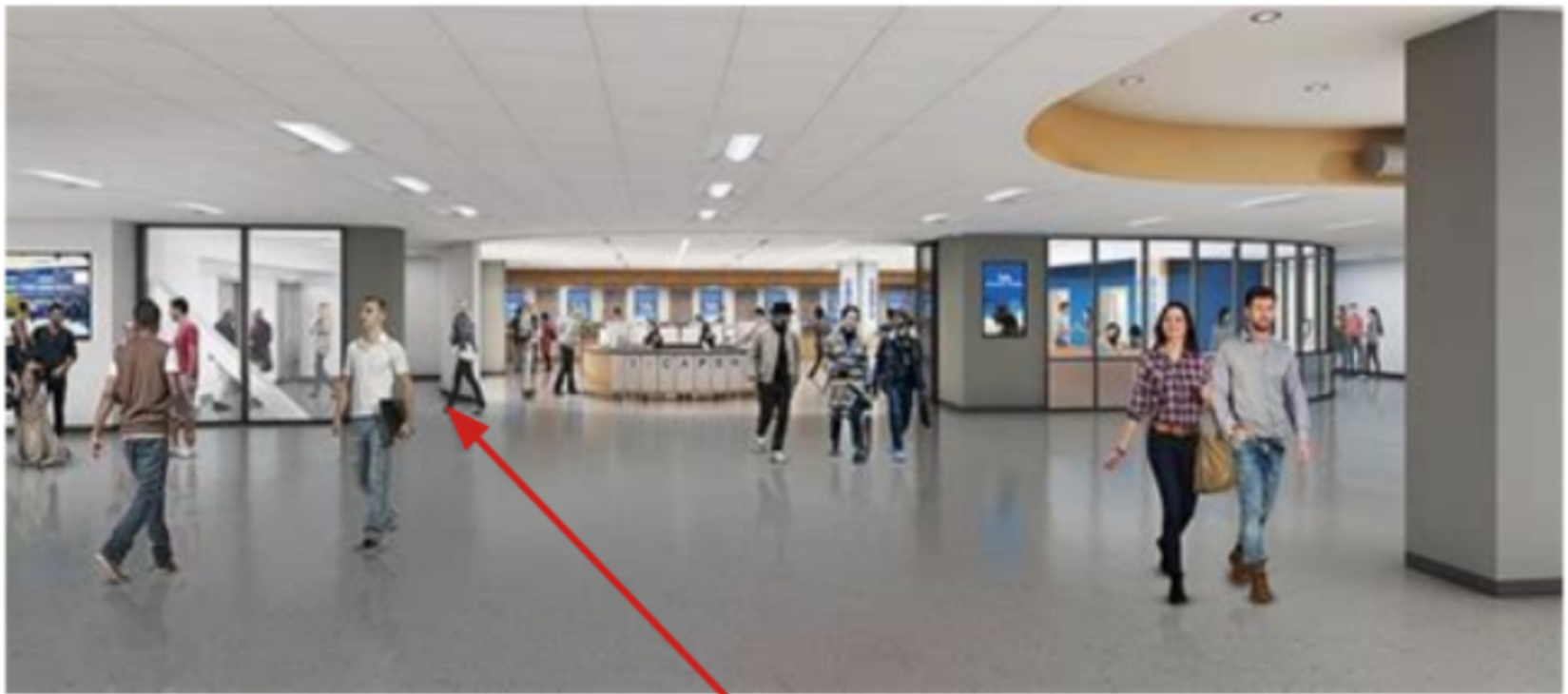
He's out this week and will be back next week.

Who are the instructors?

- Oliver Kennedy [okennedy@buffalo.edu]
  Where: Capen 212
  Office Hours: Weds 2:00-3:50
- Eric Mikida [epmikida@buffalo.edu]
  Where: Capen 212
  Office Hours: TBD

Please keep discussions on Piazza (use private posts if necessary)
Always include `[CSE-250]` on the subject line when emailing

Finding Capen 212



Who are the TAs?

TAs

- Kartike Chaurasia
- Amelia Graca
- Dikshit Khandelwal
- Nawar Khouri
- Riad Mukhtarov
- Chris Dearing
- Marian Huynh
- Derek Gage
- Ria Gupta
- Doniyor Ismatilloev
- Wonwoo Jeong
- Evan Jiang
- Ronan Kasmier
- Jonathan Guzman

TAs

- Joy Lee
- Morgan Li
- Vrushaali Nagaraj
- Brendan O'Connell
- Milos Petrovic
- Ethan Phan
- Marvin Pierre-Pierre
- Jonathan Sauter
- Shreyas Narayanan Sridhar
- Alexander Terry
- Kiki Tran
- Jordan Wang
- Eric Xie

Graders

- Sean Grzenda
- Sai Pavan Kumar Yadlapalli
- Mahima Saxena
- Shristhi Lakshmesh Karkera
- Rajesh Bammidi
- Prashant Godhwani

Staff in blue are returning from previous years.

Logistics

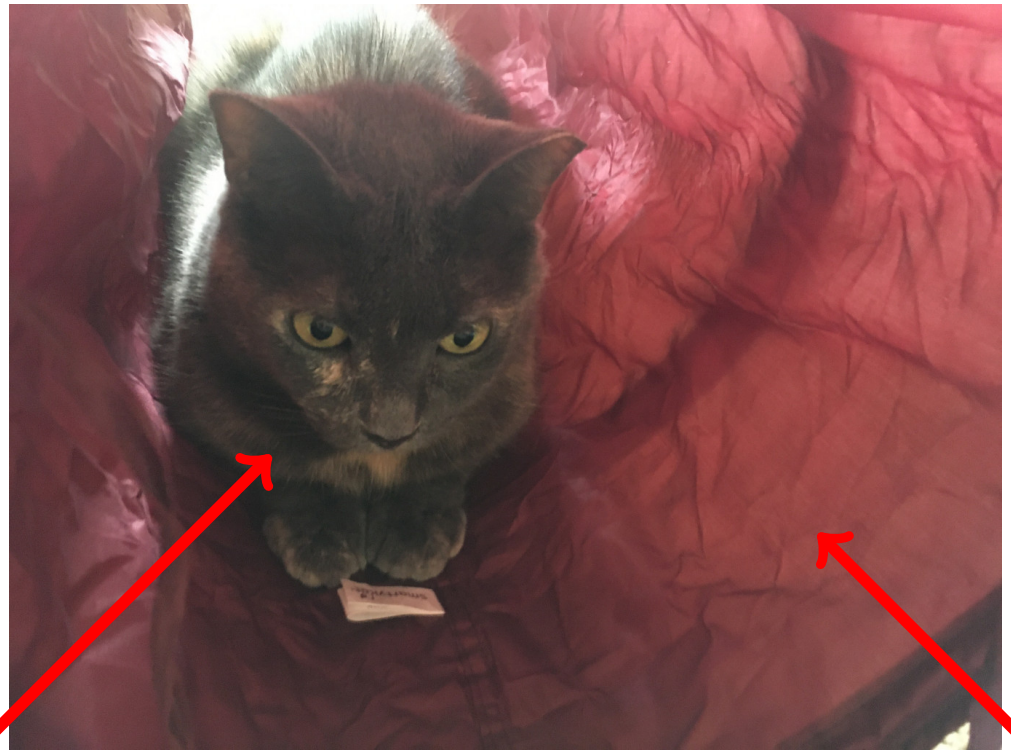
- **Course Forums + Live Q&A**: Piazza
  https://piazza.com/buffalo/fall2023/cse250
- **Course Website / Syllabus**:
  https://cse.buffalo.edu/courses/cse250/2023-fa
- **Assignment Submission**: Autolab
  https://autolab.cse.buffalo.edu
- **Assignment Distribution**: Github Classroom

Development Environment

Supported Development Environments

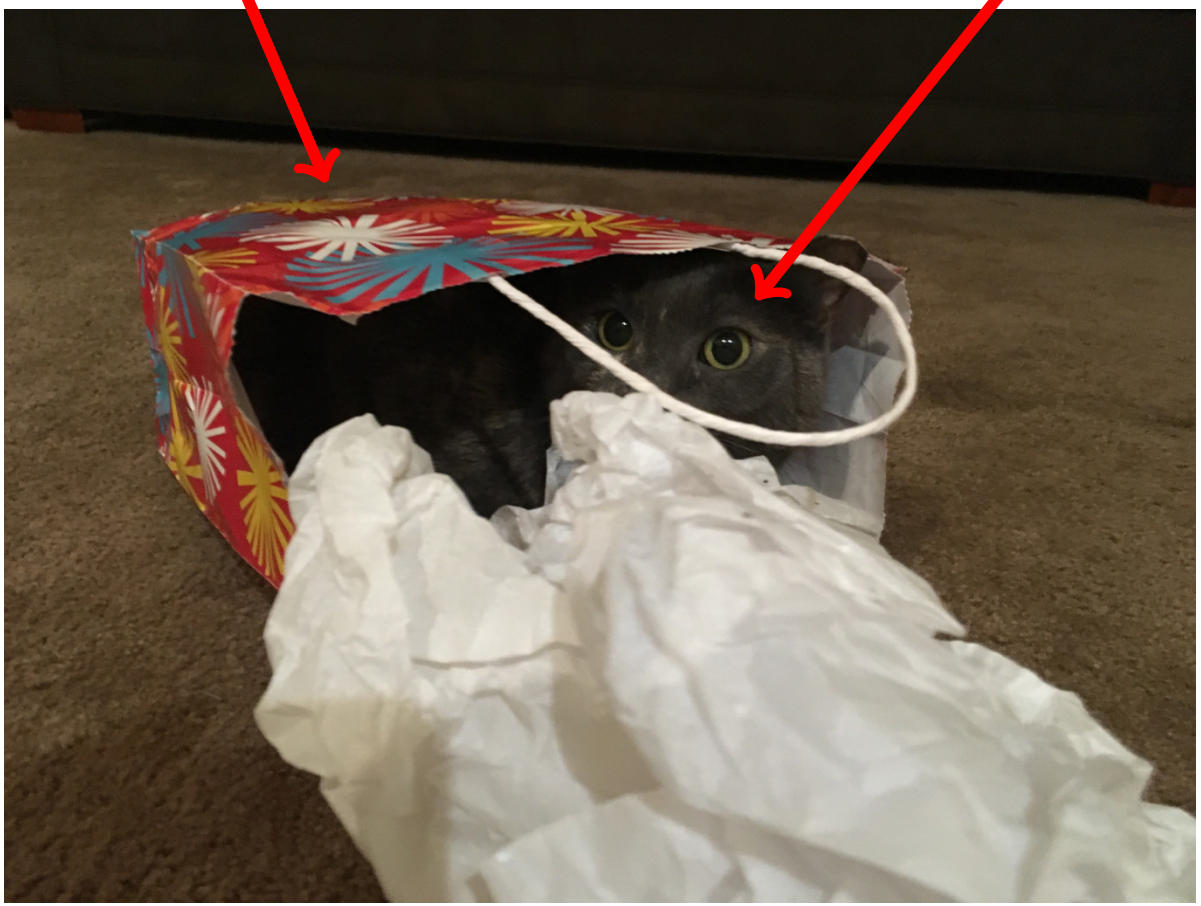
- IntelliJ

Other setups are ok, but the more your setup differs, the less we'll be able to help you.



Different Container (More Defensible)

Same Data



Different Container (More efficient `skriches()`) **Same Data**

So what is a data structure?

A **thing** to put your **things** in.

Why?

1. Which is easier to find stuff in: an organized or a messy room?
2. Which is easier to maintain?

Examples of Data Structures

- Store a list of things in some order ("List")
  - Array
  - Linked List
  - ArrayBuffer
- Store things organized by an Attribute ("Map", "Dictionary")
  - Hash Table
  - Binary Search Tree
  - Red-Black Tree

How do I make my code efficient?

- **Tactical**: Optimize your Code
  - Understand the memory hierarchy
  - Understand the CPU/OS
- **Strategic**: Optimize your Algorithm
  - Understand how your algorithm scales
  - Avoid repetition in your code

CSE 250 focuses on optimizing algorithms

Example 1

You have

- A list of UBIT / Grade pairs
- A list of UBIT / Name pairs

You want

- A list of Name / Grade pairs

Example 1

Option 1

```
    for (ubit1, grade) in grades:
      for (ubit2, name) in names:
        if ubit1 == ubit2:
          print(f"{name}, {grade}")
          break
```

Option 2

```
    name_lookup = {}
    for (ubit2, name) in names:
      name_lookup[ubit2] = name
    for (ubit1, grade) in grades:
      name = name_lookup[ubit1]
      print(f"{name}, {grade}")
```

Which is better?

Example 1

(Option 2 is called a "hash join")

($\sim$ **8 of the top 10** Fortune 500 software companies have a database product; Talk to Oliver if you're interested)

Example 2

Live Demo

(thanks to Prakshal Jain; 2021 TA for the suggestion/prototype)

C++ Standard Library

Page Discussion View Edit History

C++ Containers library std::unordered_map

std::unordered_map

Defined in header <unordered_map>

```
template<
    class Key,
    class T,
    class Hash = std::hash<Key>,
    class KeyEqual = std::equal_to<Key>,
    class Allocator = std::allocator< std::pair<const Key, T> >
> class unordered_map;
```

(1) (since C++11)

```
namespace pmr {
template <
    class Key,
    class T,
    class Hash = std::hash<Key>,
    class KeyEqual = std::equal_to<Key>
> using unordered_map = std::unordered_map<Key, T, Hash, KeyEqual,
                            std::pmr::polymorphic_allocator<std::pair<const Key,T>>>;
}
```

(2) (since C++17)

std:unordered_map is an associative container that contains key-value pairs with unique keys. Search, insertion, and removal of elements have average constant-time complexity.

Internally, the elements are not sorted in any particular order, but organized into buckets. Which bucket an element is placed into depends entirely on the hash of its key. Keys with the same hash code appear in the same bucket. This allows fast access to individual elements, since once the hash is computed, it refers to the exact bucket the element is

https://en.cppreference.com/w/cpp/container/unordered_map

Java's Util

java.util

Class ArrayList<E>

java.lang.Object
  java.util.AbstractCollection<E>
    java.util.AbstractList<E>
      java.util.ArrayList<E>

All Implemented Interfaces:

Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

Direct Known Subclasses:

AttributeList, RoleList, RoleUnresolvedList

```
public class ArrayList<E>
extends AbstractList<E>
implements List<E>, RandomAccess, Cloneable, Serializable
```

Resizable-array implementation of the `List` interface. Implements all optional list operations, and permits all elements, including `null`. In addition to implementing the `List` interface, this class provides methods to manipulate the size of the array that is used internally to store the list. (This class is roughly equivalent to `Vector`, except that it is unsynchronized.)

The `size`, `isEmpty`, `get`, `set`, `iterator`, and `listIterator` operations run in constant time. The `add` operation runs in *amortized constant time*, that is, adding n elements requires O(n) time. All of the other operations run in linear time (roughly speaking). The constant factor is low compared to that for the `LinkedList` implementation.

https://docs.oracle.com/javase/7/docs/api/

Complexity Lore

Every (good) standard data structure library provides guarantees on the complexity of its data structures' operations

Understanding complexity can be the difference between code that runs in 6 hours vs code that runs in 8 seconds.

Containers

We have

- A list of cats

We want

- To get the first cat in the list
- To add a new cat to the front of the list
- To get the nth cat in the list

This is an abstract data type

Abstract Datatypes

Stuff you store (data)

- A list of cats

Operations you can perform on the stored stuff

- To get the first cat in the list
- To add a new cat to the front of the list
- To get the nth cat in the list

So how do we store our list of cats?

Options

1. **Very Fast**: Prepend, Get First — `Linked List`
   Very Slow: Get Nth
2. **Very Fast**: Get Nth, Get First — `Array`
   Very Slow: Prepend
3. **Very Fast**: Prepend, Get First — `ArrayBuffer (reversed)`
   Sometimes Slow: Prepend

Which is best?

No one option is always best!

Data Structures

- A **Abstract Datatype (ADT)** says *what* is stored and what you can do with it.
- A **Data Structure** says *how* the stuff is actually stored/organized.

Tools

- **Specific ADTs (and Data Structures)**
  (organizational strategies)
  - Collection Types (Lists, Arrays, ArrayBuffers, Sets, Heaps)
  - Maps (Hash Tables, Search Trees)
  - Graphs
- **Algorithms**
  (recipes for common tasks)
  - Accessing/Updating Collections
  - Sorting Data
  - Graph/Tree Traversal

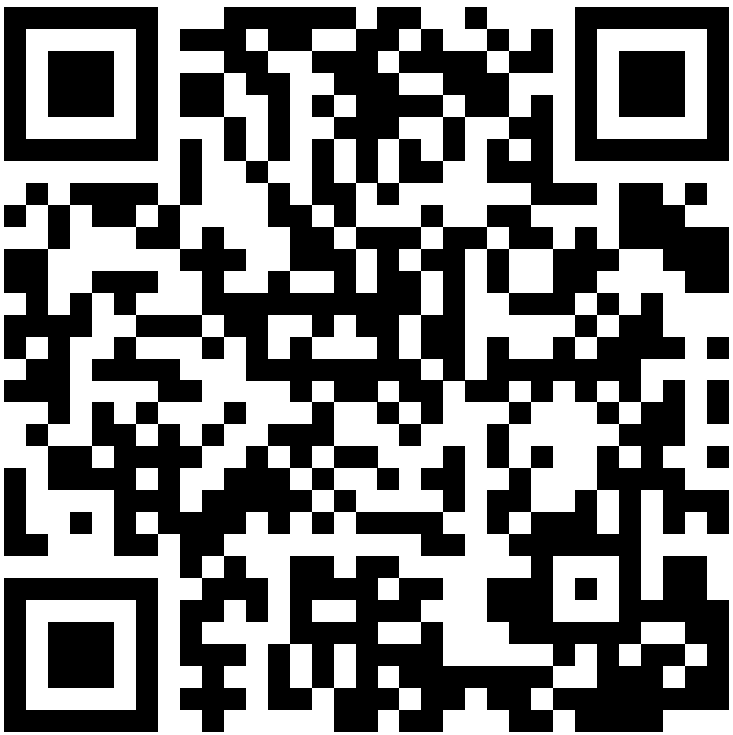
Techniques

- Pseudocode
  - Designing algorithms top-down
- Algorithm Analysis / Asymptotic Notation
  - The 50,000ft view of algorithm runtime
- Recursion
  - Expressing tasks in terms of themselves

Topic Order

- Java
- Asymptotic Notation
- Sequence Collections
- Recursion
- Graphs
- Priority Queues, Heaps
- Tree-Based Data Structures
- Hash-based Data Structures
- Advanced Topics (Secondary Storage)

Syllabus



https://cse.buffalo.edu/courses/cse250/2023-fa

Grading

Grade Breakdown

- Assignments 40% (5%. each)
- Recitation Attendance 10%
- 2 Midterms 15% each
- Final Exam 20%

Score (x)	Letter Grade	Quality Points
$90\% \leq x \leq 100\%$	A	4
$85\% \leq x < 90\%$	A-	3.67
$80\% \leq x < 85\%$	B+	3.33
$75\% \leq x < 80\%$	B	3
$70\% \leq x < 75\%$	B-	2.67
$65\% \leq x < 70\%$	C+	2.33
$60\% \leq x < 65\%$	C	2
$55\% \leq x < 60\%$	C-	1.67
$50\% \leq x < 55\%$	D	1
$0\% \leq x < 50\%$	F	0

Written Assignments

- **Bi-Weekly Written Assignments**
  Expect to spend about a week working on it
  Submissions allowed up to a day late (50% penalty)
- **You are responsible for submission format**
  Submit **only** PDFs
  Submissions that can't be read will receive a 0
- **We recommend writing solutions by hand**
  Handwritten work is retained more effectively
  It's easier to write out math by hand

Programming Assignments

Typical Grade Distribution

- Write Test Cases (∼15/100 points)
  Submit as many times as you like
- Test Submission (50/100 points)
  Submit as many times as you like
- Final Tests (15/100 points)
  Tests are run exactly once, after the deadline passes

Your grade is based on your <u>most recent</u> submission.

Assignments

Assignments are released on **Mondays** and due on **Sundays** at 11:59 PM.

Course staff have lives (yep, it's true).
Do not expect help after 5 PM on the Friday before it is due.

Exams

- In-Class Midterm 1 (**October 2**)
  - Content Coverage is roughly Weeks 1-5 in the syllabus
  - More details prior to the exam
- In-Class Midterm 2 (**November 10**)
  - Content Coverage is roughly Weeks 6-11 in the syllabus
  - More details prior to the exam
- One Final Exam (**Tuesday December 19, 2023; 3:30-6:30**)
  - Comprehensive exam (all topics are fair game)
  - Determine if you have a conflict ASAP
  - If HUB updates, trust the date in HUB

Please contact **Accessibility Resources** for accommodations

https://www.buffalo.edu/studentlife/who-we-are/departments/accessibility.html

Attendance / Participation

- **Lecture**
  - No recorded attendance (unless you make us)
  - You are paying $$ to be able to ask questions live (don't waste it)
- **Recitation**
  - Recitations start Tue, Sept 5 (Next week)
  - Attendance is mandatory

Collaboration

- Do...
  - ... work together to brainstorm ideas
  - ... explain concepts to each other
  - ... include a list of collaborators on all submitted work
- Do **not**...
  - ... write solutions while working together
  - ... describe the details of solutions or code
  - ... leave your code in a place where someone else can see it

If in doubt, ask a member of the course staff.

Resource Policy

- Do...
  - ... use materials provided by course staff (Piazza, Class, OH)
  - ... use materials provided by textbooks, readings
  - ... cite materials you reference for written work
  - ... cite sources for all code you reference/copy

Resource Policy

- Do **not**...
  - ... reference random videos that "helped you solve the problem"
  - ... reference exact solutions found online
  - ... use chatbots
  - ... hire private tutors

Why?

- This is an intro-level course. Almost nothing you learn here is "new".
- If you don't understand, you will struggle with later courses (e.g., 331).
- If someone else does the work, you're not the one that understands.
- We want you to understand the pieces, so that you can (eventually) start fitting them together in clever ways.
- If we catch you cheating, **you get an F**.

Other Ways to Get an F

- Work in a group by assigning each person to a problem.
- Copy a friend's homework because you forgot ($\sim$1% of your grade is not worth it)
- Share your homework with your friend (I can't tell who copied)
- Submit work without citations (Cited work included in your project is not an AI violation)
  (Although we will grade you on the work you did)

You are liable if someone else submits your work as their own.

Amnesty Policy

You may retract any work you submit, *at any time* before we discover that it contains an AI violation.

```
Dear Dr. Kennedy,

In order to preserve academic integrity in CSE
250, I would like to withdraw my submission for
Project/Homework XXX.

Sincerely,
  Name (ubname@buffalo.edu)
```

Life Lesson

If ChatGPT can do your work... you will not be employable for very long.

When to Ask Questions

- In Class (raise your hand[1])
- Piazza (Ask anytime!)
- Office Hours (All the TAs have been where you've been)
- Recitations (... if you prefer smaller, less intimidating settings)

[1] Oliver's slides always have small mistakes 'to make sure that you're paying attention'.

How to Ask Questions

A good question should include...

- What is your goal/objective?
  (Avoid phrases like "it's not working")
- What did you try?
  (Try to solve the problem yourself before asking)
- How did the things you tried break?
  (It's not always obvious)
- Text of code/error messages, and *not screenshots*
  (Piazza can't search images)

AI Quiz

Log into autolab; it will take you under 10 minutes.

Due Weds, Sept 6 at 11:59 PM

Successfully completing the AI exam with a passing grade is mandatory. If it is not complete by the deadline, you get an 'F'.

Java Hello World Project

Posted on course website; Submit a java program that prints out your github username.

Due Sun, Sept 10 at 11:59 PM

This project does not count for a grade, but submission is required by the deadline to pass the class.

Questions?