CS 261-020
Data Structures

Lecture 1

Introduction and Course Syllabus

1/9/24, Tuesday

Oregon State University

Odds and Ends

- We have recitations this week
  - Recitation 1 posted on Canvas
  - Go to your registered recitation

- Assignment 1 posted

Lecture Topics:

- Course Intro
- Syllabus
- C Basics

Course Intro

- *"… the difference between a bad programmer and a good one is whether [s]he considers his[/her] code or his[/her] data structures more important. Bad programmers worry about the code. Good programmers worry about **data structures and their relationships**."*

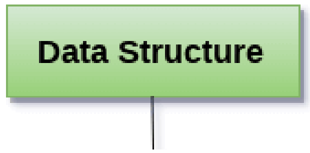
  -Linus Torvalds, creator of the Linux kernel

Data Structure

- Data structures are general-purpose mechanisms for storing, organizing, and managing data within a running program.
    - Encapsulates the operations associated with a particular structure

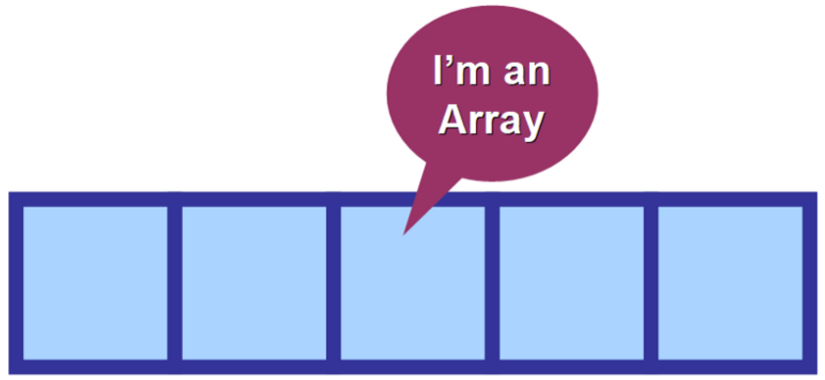
- a given data structure represents not only the stored data itself, but also often represents the **relationships** between specific data elements

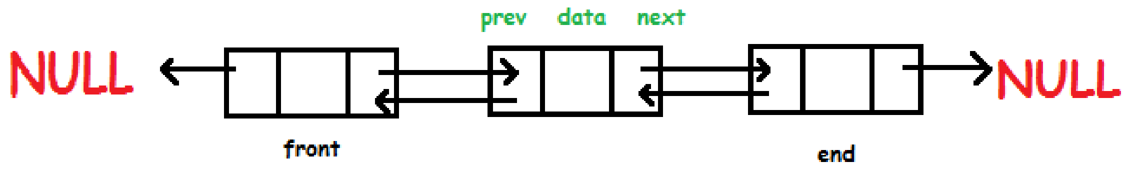
Data Structures Classification

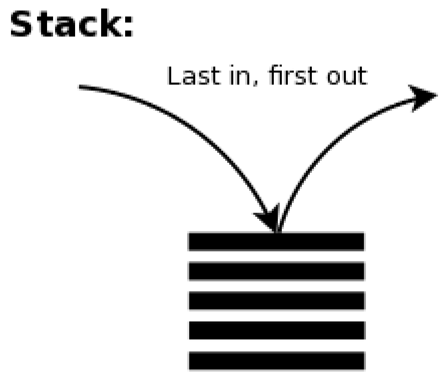


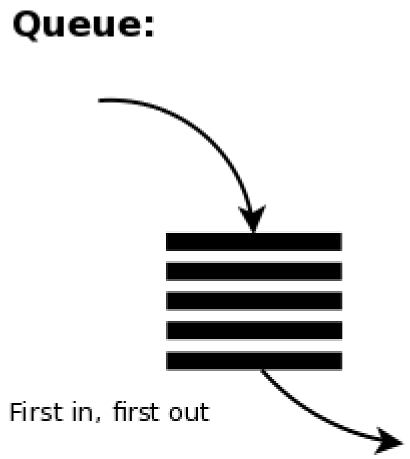
https://www.javatpoint.com/data-structure-introduction

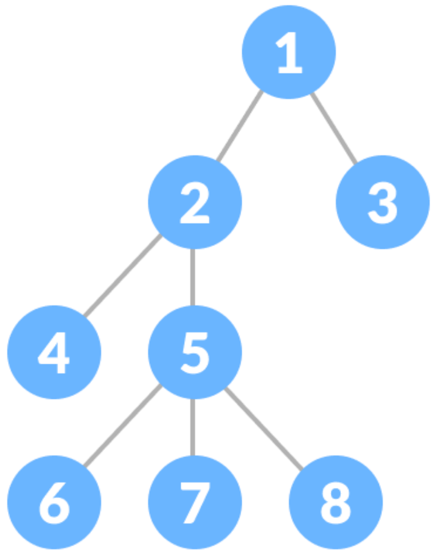
Data Structure Examples

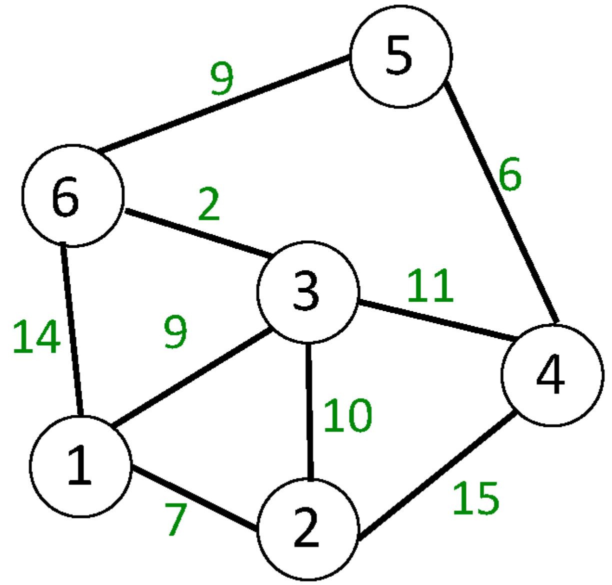










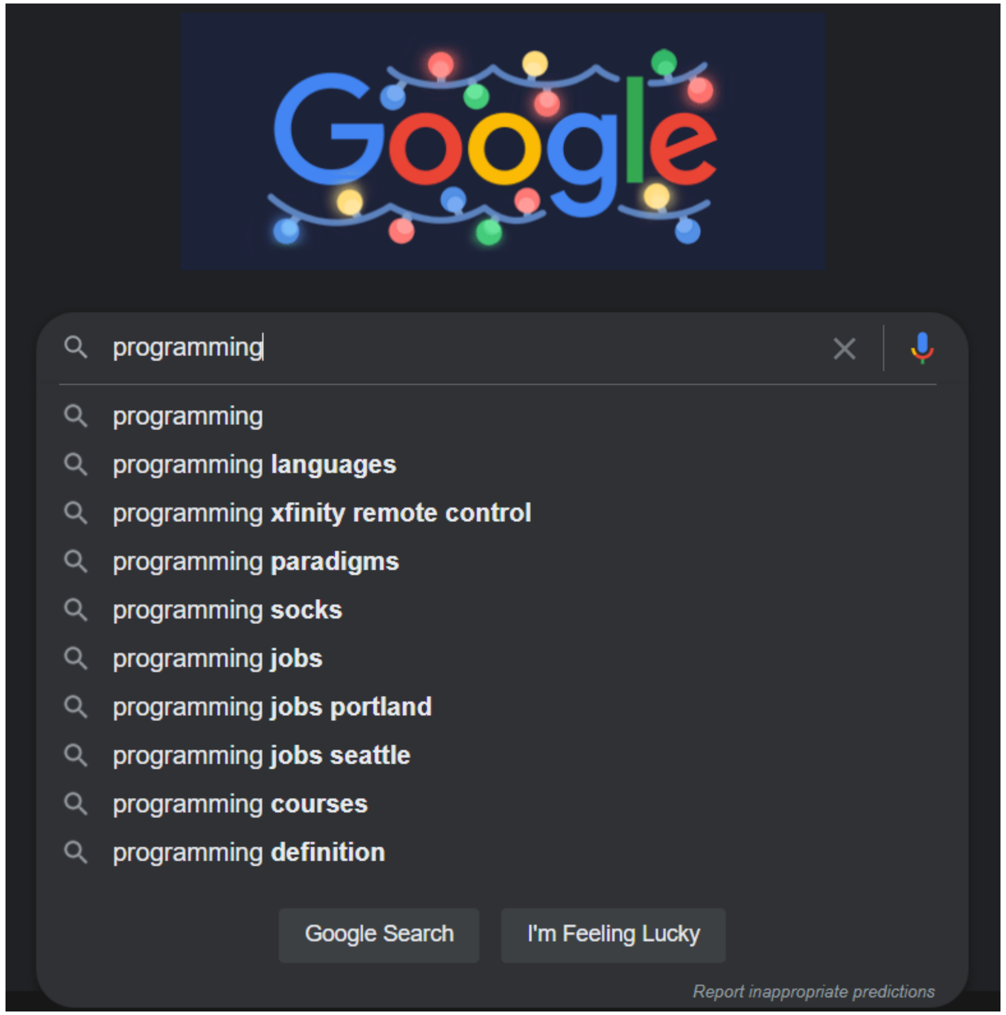


A real-life example: Auto-complete

- Scenario: You are asked to add an autocomplete feature to the search box in your web application. This feature will behave much like Google's autocomplete feature.

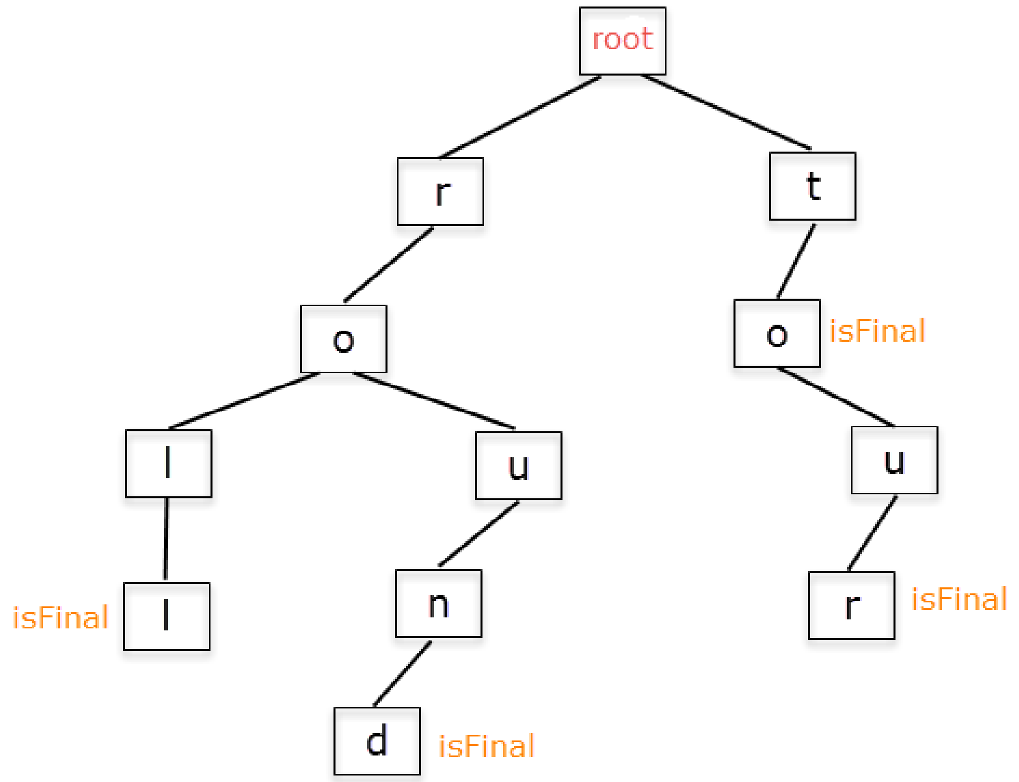
- The data for this feature is already compiled and provided to you in an alphabetically-sorted text file that contains one completion per line.

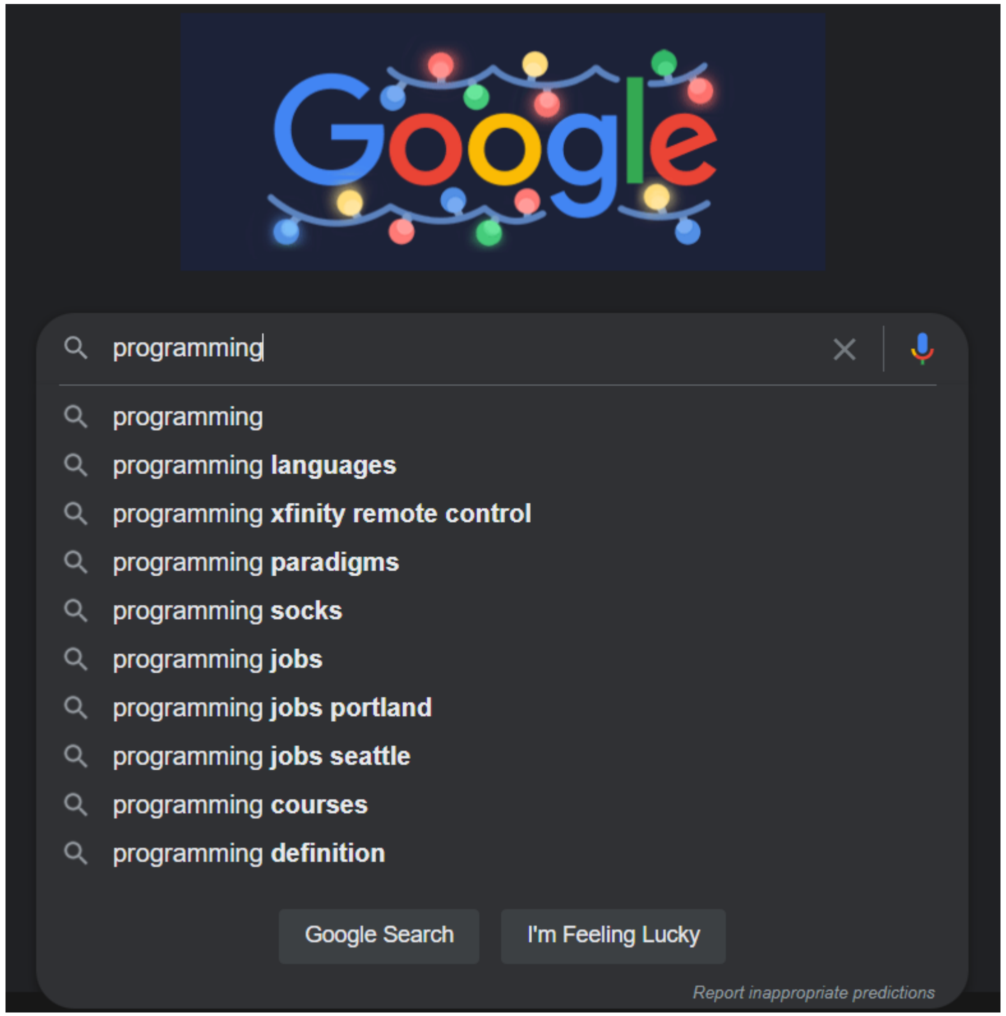
- Question: How are you going to store and use that data in your running web application?


programming
programming
programming languages
programming xfinity remote control
programming paradigms
programming socks
programming jobs
programming jobs portland
programming jobs seattle
programming courses
programming definition

A real-life example: Auto-complete

- One possible solution: Trie (pronounced as "try")
  - also called **digital tree** or **prefix tree**




programming

programming
programming **languages**
programming **xfinity remote control**
programming **paradigms**
programming **socks**
programming **jobs**
programming **jobs portland**
programming **jobs seattle**
programming **courses**
programming **definition**

Goals:

- Be familiar with a collection of foundational data structures
  - dynamic arrays, lists, queues, stacks, trees, hash tables, graphs, etc.
- To understand how to analyze and manage the complexity associated with data structures and their operations
  - Gives more control to our programs' running times and memory usage
- Be able to compare data structures and choose/design the best one for a particular task

Caveat

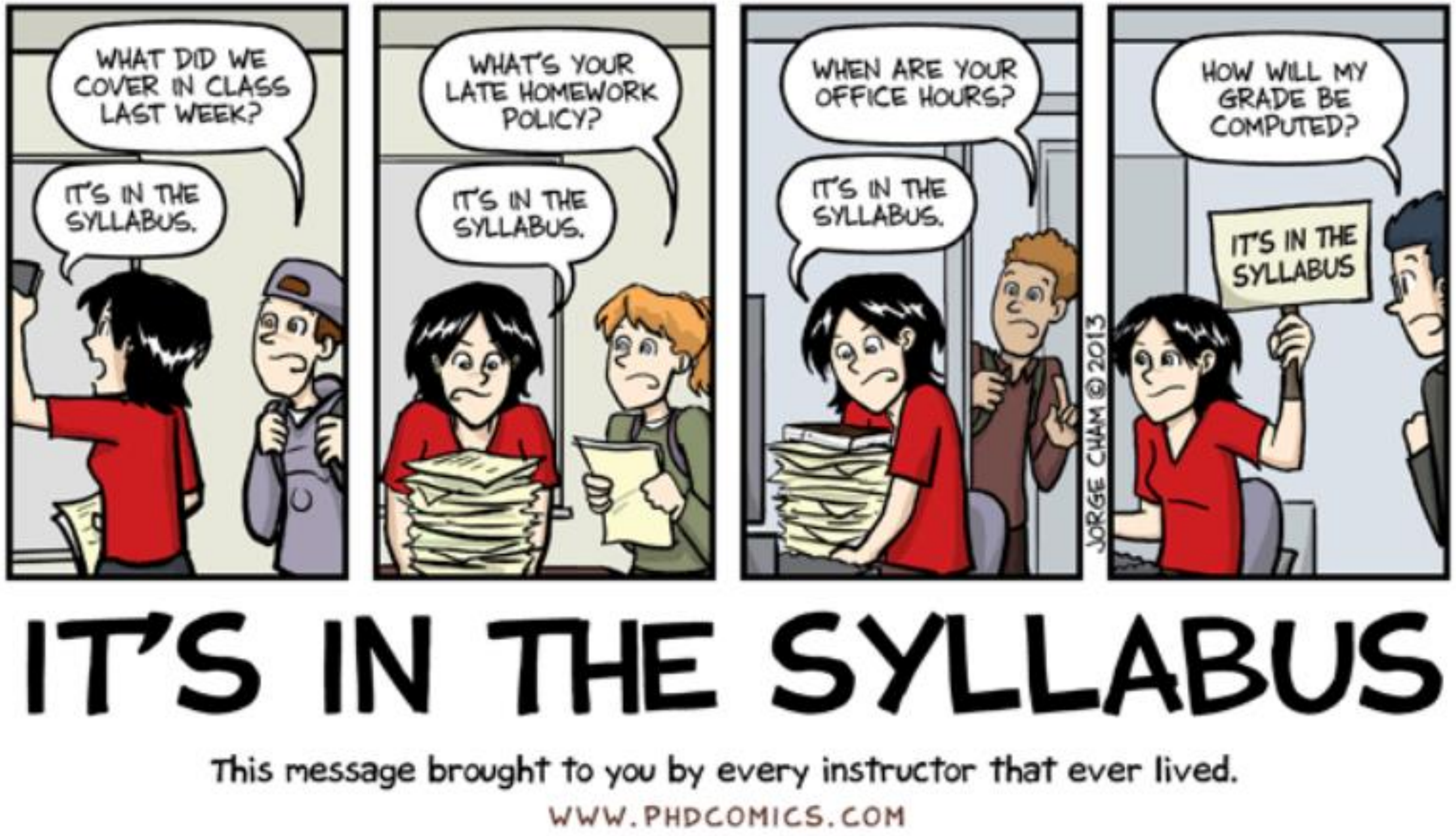
- None of the data structures is a perfect data structure for all situations!

- Things to consider…
  - How long does it take to run? (time)
  - How much space does it require to store the data of given size? (space)
  - How hard is it to implement?

Lecture Topics:

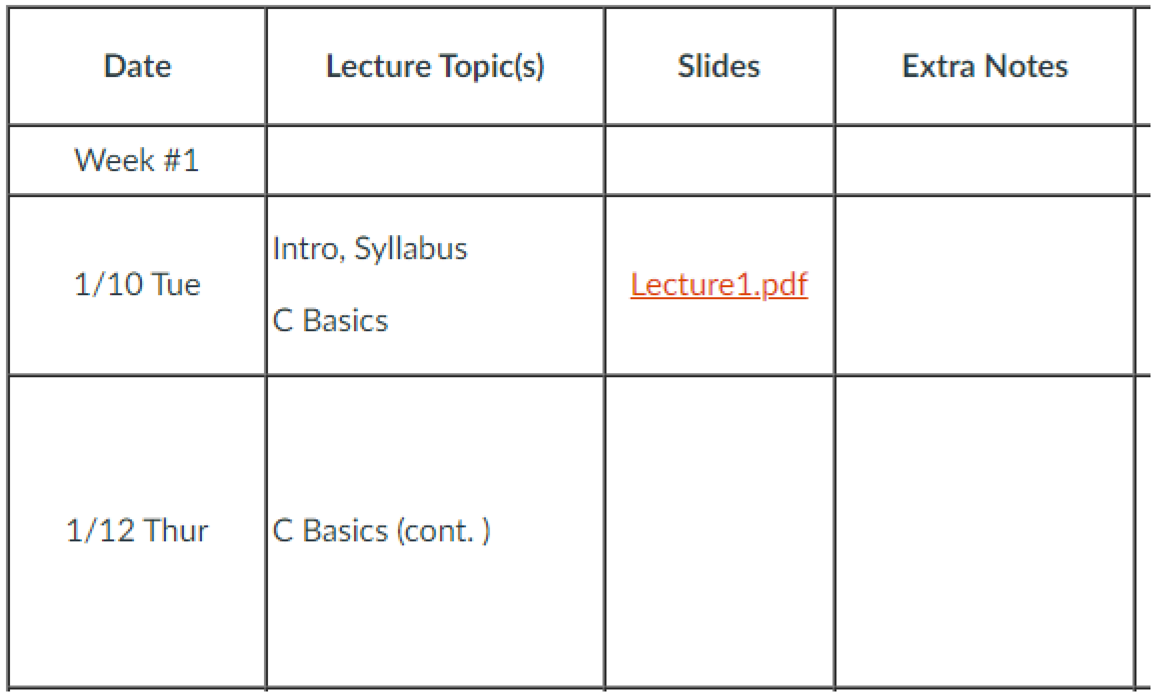
- Course Intro
- Syllabus
- C Basics

Syllabus


WHAT DID WE COVER IN CLASS LAST WEEK?
IT'S IN THE SYLLABUS.
WHAT'S YOUR LATE HOMEWORK POLICY?
IT'S IN THE SYLLABUS.
WHEN ARE YOUR OFFICE HOURS?
IT'S IN THE SYLLABUS.
HOW WILL MY GRADE BE COMPUTED?
IT'S IN THE SYLLABUS
JORGE CHAM ©2013
IT'S IN THE SYLLABUS
This message brought to you by every instructor that ever lived.
WWW.PHDCOMICS.COM

Course Structure

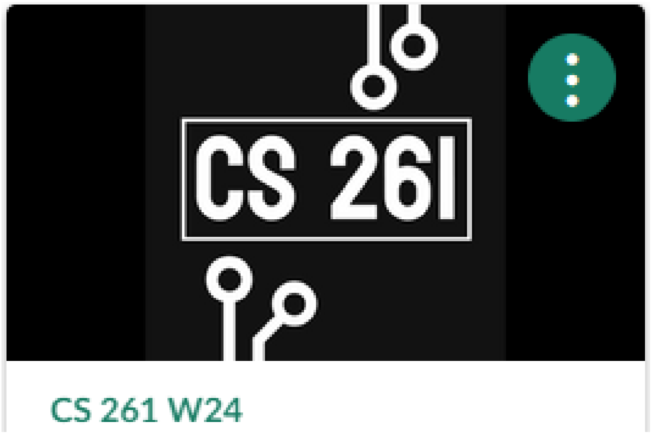
- 10 weeks schedule
  - Weekly Schedule on Canvas (Calendar page)
  - C Basics (Week 1)
  - Array and list (Week 2)
  - Complexity Analysis (Week 2-3)
  - Stack, queue, deque (Week 3-4)
  - Trees (Week 5-7)
  - Priority queues, heaps (Week 7-8)
  - Map and Hash Table (Week 8-9)
  - Graph (Week 9-10)

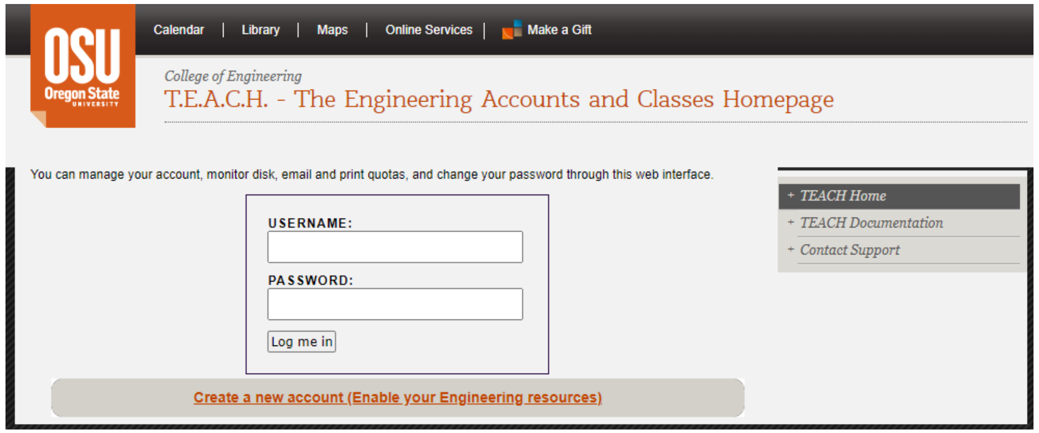


Date	Lecture Topic(s)	Slides	Extra Notes
Week #1			
1/10 Tue	Intro, Syllabus C Basics	Lecture1.pdf	
1/12 Thur	C Basics (cont.)		

Course Information

- Canvas site:
  - All course materials
- TEACH:
  - Code submission (as .c)
- Discord:
  - Online discussion and Q&A forum



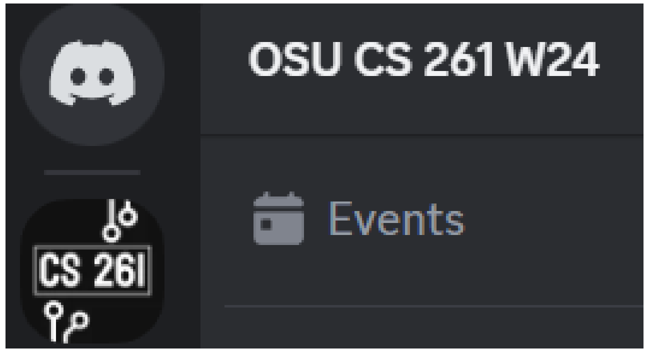

OSU Oregon State University
Calendar | Library | Maps | Online Services | Make a Gift
TEACH Home
TEACH Documentation
Contact Support
USERNAME:
PASSWORD:
Log me in

College of Engineering

T.E.A.C.H. - The Engineering Accounts and Classes Homepage

You can manage your account, monitor disk, email and print quotas, and change your password through this web interface.

Create a new account (Enable your Engineering resources)



Basics

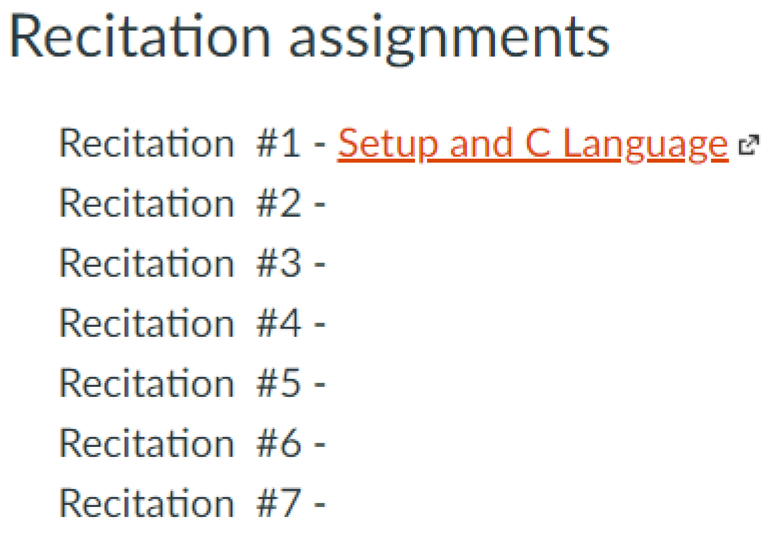
- Instructor: Yipeng (Roger) Song
  - I go by Roger ☺
- Email
  - Instructor: songyip@oregonstate.edu
  - TAs: cs261-ta@engr.orst.edu (TAs and me)
- Office Hours: TBD @ TBD
- Requirements: Laptop (Windows, MacOS, or Linux)
- Programming Language: C

More Basics…

- Be respectful (Establishing a Positive Community)
- Have a growth mindset
  - Most abilities could be developed through dedication and hard work
- Don’t cheat (0 tolerance!!)
- Be Proactive
  - Take control and cause something to happen, rather than just adapt to a situation or wait for something to happen

Attendance

- **Lecture**: Strongly Encouraged
  - I will post lecture slides, demoed code, and additional resources on Canvas → Calendar
  - You are expected to be present during exam dates!!!
- **Recitation**: Required
  - Recitation 1 document is posted on Canvas → Recitations
- Missed recitations result in a zero for that recitatio
  - Email TA mailer BEFORE the end of recitation
  - Subject: “[CS261-020] Missing a Recitation”
  - Recitation you are missing
  - Excuse for missing recitation
  - Plan for making up the recitation


Recitation assignments

Recitation #1 - Setup and C Language

Recitation #2 -

Recitation #3 -

Recitation #4 -

Recitation #5 -

Recitation #6 -

Recitation #7 -

Grade Breakdown

- 20% - Recitations
- 40% - Assignments
- 10% - Bi-Weekly Quizzes
- 30% - Exams
  - 15% - Midterm
  - 15% - Final

Recitations – 20%

- 10 in total
  - Recitation materials will focus on implementing topics from class

- 10 pts per recitation, correctness + effort-based, check off with your recitation TAs during recitation time to get points
  - Do not leave unless being checked off
  - Submit your recitation work to TEACH for backup purposes

- You MUST attend the recitation in which you registered (unless you received permission from the TAs or me)

Assignments – 40%

- 5 in the term
- Two-week assignments
- **Always something due Sunday by midnight**
- **All code must compile on ENGR – otherwise 0 (coding portion)**
- Late Policy (only for coding portion!!!)
  - 1 day late: 10% penalty
  - 2 days late: 30% penalty
  - 3 days or more: not accepted → 0
  - No grace days…

Assignment Grading

- Assignment 1-4 are demoed (in person)
- Assignment 5 will be graded by the TAs on their own during final's week
- Sign up for a demo for assignment 1-4 using links on TA Hours page on Canvas
- Demo within 2 weeks of the code due date, even if late
    - Missing a demo, **-10 pts**
    - Demoing outside 2 weeks w/o permission, **-30%**

- **Assignments that are not demoed at the end of the term → 0 pts**

Bi-Weekly Quizzes – 10%

- Due every other Sunday midnight (5 in total, on Canvas)
- Available from: after 2nd lecture to Sun 11:59 pm
  - Canvas is very unforgiving about due times -- don't push it.
- T/F, and multiple choices, covering materials taught in that week
- 5 to 10 questions on each quiz, with a 60-minutes time limit
- 2 attempts for each quiz, keep the highest score

Look at the bi-weekly:

Mon	Tue	Wed	Thur	Fri	Sat	Sun
						1) Asm `N` Due
	Lecture		Lecture			1) Quiz `N+1` Due
	Lecture		Lecture	Asm `N` Demo Due		1) Asm `N+1` Due

Exams – 30%

- Mid Terms – 15%
  - Week 5 Tuesday (Feb 6)
- Final – 15%
  - Final's Week: Wednesday 2:00 pm (Mar 20)

- Non-cumulative (but it builds on…)
- Same classroom

Grading Philosophy*

A	[93 or greater) mastery
A-	[90 – 93)
B+	[87 – 90)
B	[83 – 87) stable/proficient
B-	[80 – 83)
C+	[77 – 80)
C	[73 – 77) passable
C-	[70 – 73)

*Note: I do roundings ☺ (i.e. 89.45 → 89.5 → 90 ☺)

How to Be Successful

- Read and listen carefully
- Start assignments early
- Be proactive with absences and issues that arise in the term
- Get help when you need it
  - Make use of Discord and Office Hours

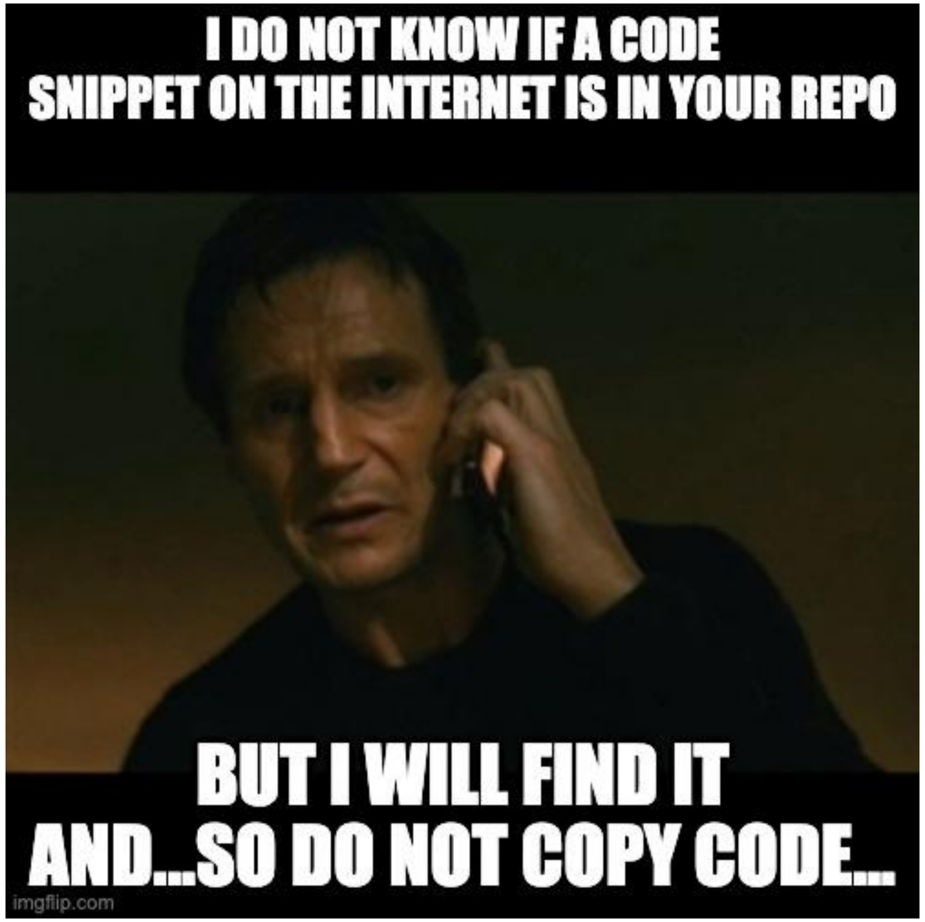
Recitation and Assignment Rules

- DO NOT SHARE YOUR WORK OR CODE WITH OTHER STUDENTS
  - **You are encouraged to discuss with others** about the assignments but do not ask/give your work to the others
  - **Do not copy** other students' work or resources available (without citations) in online
  - **Do not publish** your work online

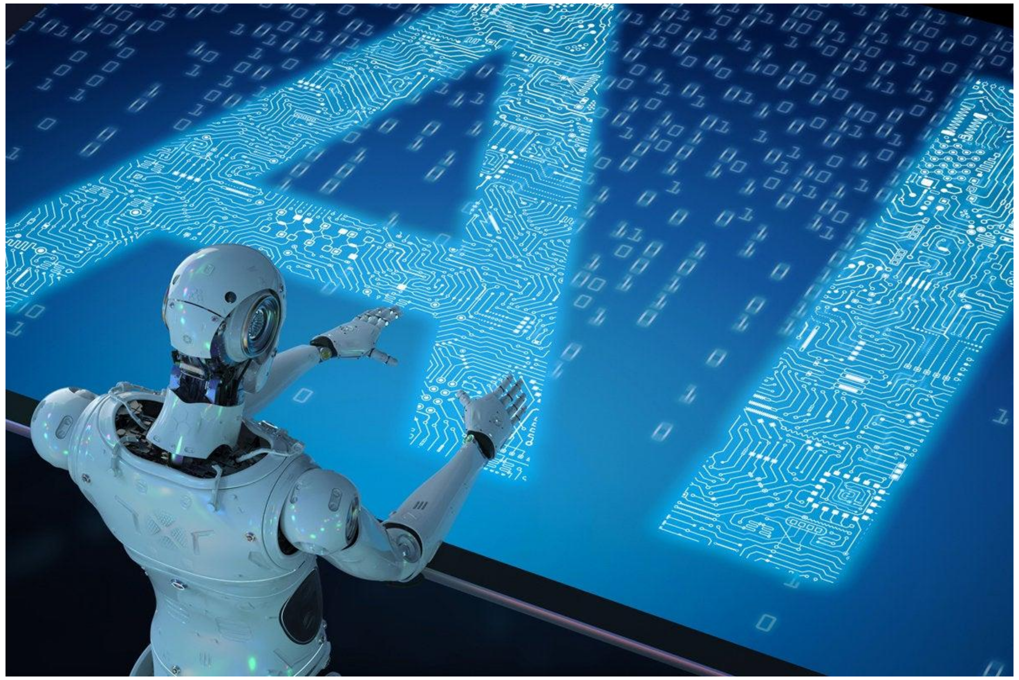
Recitation and Assignment Rules

- Plagiarism will be punished via the Office of Student Life..
  - E.g., getting F or zero point for the recitation/assignment that matters with plagiarism…

- Please refer the Code of Student Conduct


I DO NOT KNOW IF A CODE SNIPPET ON THE INTERNET IS IN YOUR REPO
BUT I WILL FIND IT
AND...SO DO NOT COPY CODE...
imgflip.com

AI Tool Usage in this class



- You must be the author of **all work**
- You may use AI to:
  - generate abstract ideas
  - polish or edit text you have drafted
  - quiz yourself
  - explain new or confusing concepts
  - generate code snippets to solve **unassigned example tasks**
- You may **NOT** use AI to
  - generate code snippets to solve a problem presented in a quiz, recitation, assignment, or exam
  - draft the code implementation for an assignment
- If used, add a citation just like you would when you copy language or code from human authors.

Tips to the Recitations/Assignments

- Study in a group (discussion is highly encouraged!)
  - But please **write code individually**!
- Read the document thoroughly and follow the instructions
- Ask questions (Discord)
- Understand your time budget
  - **Plan ahead** to finish the recitations/assignments on time

TAs

- Go see your TAs!!!
- Where: Varies
- When: Varies – check the TA Hours page on Canvas

Help Hierarchy

- Reread assignment, lecture slides, recitations, syllabus
- Google/Bing/Open a textbook
- Ask a friend
- Check Discord for relevant posts or create a new question
- Ask a TA
    - You can attend office hours in person
    - TAs will also be monitoring Discord
- Ask Roger

Lecture Topics:

- Course Intro
- Syllabus
- C Basics

C Basics

- Programming language: C
  - C99 standard of the C language

- Compiler: GCC (installed on ENGR server)
  - E.g. Compile a single C file (`main.c`) using the GCC C compiler (under the C99 standard) to produce an executable file (`main`):

    ```
    gcc -std=c99 main.c -o main
    ```

  - `-std=c99` allows declaration of variables anywhere in a block, otherwise, C language forces to declare all the variables at the beginning of a block

C Basics – C Program Structure

- `main()` function: -- entry point into the program
- Include statements at the top of the file
  - The standard file extension for header files in C is .h
- No `using namespace std;` anymore

```
#include <stdio.h> //standard I/O, writing to / reading from the console/file


int main(int argc, char** argv) {
        return 0;
}
```

C Basics – printf()

- `printf()` – Print text to stdout (standard output stream)
  - In C++, we use `cout`
  - In C, we use `printf()`
    - `printf("This is a string I’m printing to stdout.\n");`

C Basics – printf() (cont.)

- How to print the content of a variable?
  - Passing a **format string** and accompanying arguments to `printf()`
    - *Format string*: a template for the text to be printed. Contains format specifiers into which specific value will later be inserted
    - *Format specifier*: start with a %, followed by a character describing the data
  - E.g.:

```
int x = 8;
printf("This is the value of x: %d\n", x);
```

C Basics – printf() (cont.)

- Common format specifiers:
  - `%d` – indicates an `int`, to be printed as a signed decimal number
  - `%f` – indicates a `double`, to be printed in fixed-point notation (e.g. 3.1415…)
    - `float` arguments are cast as `double`
  - `%c` – indicates a `char`, to be printed as a readable character
  - `%s` – indicates a null-terminated string
  - `%p` – indicates an address (or pointer)
  - Lots more…

C Basics – printf() (cont.)

- Print multiple values
  - By inserting multiple format specifiers:
  - E.g.

```
char* name = "Luke Skywalker";
double gpa = 3.75;
printf("%s's GPA is %f\n", name, gpa);
```

C Basics – scanf()

- How to accept input from standard input (keyboard)?
  - In C++, we use `cin`
    - i.e., `cin >> var;`
  - In C, we use `scanf()`
    - i.e., `scanf("%d", &var);`

  - To read in more than one value, use multiple format specifiers
    - i.e.,

```
printf("Enter two integers: \n");
scanf("%d %d", &var1, &var2);
```

C Basics – If/else and switch statements

- Similar to C++

```
if (a == 0) {
   /* Do something. */
}
else if (b != 0) {
   /* Do something different. */
}
else {
   /* Do a third thing altogether.
*/
}
```

```
switch(grade) {
        case 'A' :
                printf("Excellent!\n" );
                break;
        case 'B' :
        case 'C' :
                printf("Well done\n" );
                break;
        case 'D' :
                printf("You passed\n" );
                break;
        case 'F' :
                printf("Better try again\n" );
                break;
        default :
                printf("Invalid grade\n" );
}
```

C Basics – Loops

- Similar to C++
    - for, while, do-while

```
while (i != 16) {
  /* Do something repeatedly until i is 16. */
}
```

```
int i;
for (i = 0; i < 32; i++) {
  /* Do something 32 times. */
}
```

```
do{
  /* Do something repeatedly until i is 16. */
}while (i != 16);
```

C Basics – Functions

- No Class or Class functions

```
#include <stdio.h>

/* This could be in a separate .h file too */
void foo(int);

/* This could be in a separate .c file */
void foo(int x) {
        printf("foo was passed this argument: %d\n", x);
}

int main(int argc, char** argv) {
        foo(2);
}
```

C Basics – Functions (cont.)

- Unlike C++, C has no reference types!
- Can only pass by value (or by pointers)

```
#include <stdio.h>

void foo(int *x) {
        printf("foo was passed this argument: %d\n", *x);
}

int main(int argc, char** argv) {
       int val = 5;
       foo(&val);
}
```

To-dos before next lecture

- Read through the syllabus
- Start the recitation 1