Agenda for Today

1. Course Syllabus
2. First Assignment
3. Code Quality
4. Stack Implementation
5. Wrap Up
Course Syllabus
Syllabus Highlights

Discuss responses to the following questions with your neighbor(s).

Who can I go to for help with labs? What about assignments?

What is the course policy on GitHub Copilot?

What should I do if I need to miss class or two?
Syllabus Highlights (responses)

Who can I go to for help with labs? What about assignments?
Labs: your partner, office hours with instructor or mentor.
Assignments: *no other students*. Mentors and instructor are fine.

What is the course policy on GitHub Copilot?
Don't use it!

What should I do if I need to miss class or two?
Let the instructor know ahead of time, if possible.
If you missed a lot of days, make sure you're participating on Teams.
Ask another student from class, come to office hours, review notes from class, do the reading.
What are the "journals?"

They'll take the place of coding assignments later in the semester.
Assignment 1: Ngram Generator
Assignment Basics

Collaboration
You are expected to work independently on all assignments for this course. You can use online resources if you cite them, but don't copy–paste solutions. Mentors or I can help answer questions about assignments.

Working
I generally expect you to do your work for this class on MathLAN. If you work on your own computer I might not be able to help you set it up.

Submitting Work
Upload your assignment to Gradescope. Wait for the autograder to finish. You can submit as many times as you want up to the deadline.
Quick Demo
Go to gradescope.com, log in or set up an account, and use the code below to join the course.

G23BYN
Code Quality
Code Quality Discussion

Discuss the question below with your neighbor(s)
What makes for "high quality" C code?
What makes for "quality" C code?

Documentation: explain what each code snippet does. Explain what each function does. If variables are confusing, explain their purpose.

Memory: when you allocate memory with malloc, free it when you are done.

Readability: use good, descriptive names for variables. Think about the structure of your code. Try not to write functions that are too long. Be concise.

Performance: mostly not important, but avoid unnecessary work. Avoid unnecessary memory allocation (use locals whenever possible).

Formatting: Be consistent with code style.
C Practice: Stack Implementation
Writing a Stack in C

Discuss the following questions with your neighbor(s).

What are the parameters and return types for each operation?

How will we store the stack's contents? Think about data types we will need.
Writing a Stack in C  *(responses)*

What are the parameters and return types for each operation?

```c
int stack_pop(int_stack_t* stack)
void stack_push(int_stack_t* stack, int value)
int stack_peek(int_stack_t* stack)
void stack_init(int_stack_t* stack)
void stack_destroy(int_stack_t* stack)
```

How will we store the stack's contents? Think about data types we will need.

- An array: more efficient use of space
- A linked list: easy/efficient to append
Live Coding
Reminders

Reading
Read three chapters about processes for Wednesday's class.

Assignment
The ngram assignment is due next Monday at 11:59pm