Agenda for Today

1. First Assignment
2. Syllabus and Assignment Q&A
3. What is "good" C code?
4. Debugging Exercise
5. Wrap Up
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
Syllabus Q&A
Assignment Q&A
What is "good" C code?
Comments

Where should we add comments in our code? What should they tell us?
Put comments in places where your "community" expects them.

Before all functions: what is the function for, what it expects from parameters

In front of blocks of code like loops, if/else, etc.: What is going on?
  What does the loop check for, what does the if/else check for?
  In a large if/else sequence, explain what each case does.

Near variables: make sure it's clear what variables are for, why they are used.

Citations: cite any sources you use other than our text, class, or man pages

Any other place where the purpose of the following code is not obvious.

Consider: Somewhere early in the program that says what the program is
Formatting

How should our C code be formatted? Why?
Try not to write especially long lines.

Indent code consistently (graded). You can use a style that you like, or the default in the VSCode projects I distribute.

Include files at the top of your source:
Include everything you use. Do it at the top so it's obvious.

Write functions at the top level only (no functions inside of main)

Use curly braces for all loops and conditionals. The one exception is a stand-alone "if" where the consequent appears on the same line.
Other Rules

What other rules should we follow to make sure we're writing "quality" code?

Name variables consistently (CamelCase, lowerCamelCase, snake_case, ...)

Consider declaring functions at the top of your source:
   Makes it clear what the functions are, and lets you organize definitions

Break code into functions: helps readability, and allows code reuse

Avoid redundant code: try not to duplicate things when you can avoid it

Free memory you malloc (graded): allows memory reuse. It's also a good practice.

Give relevant names for things: names should be descriptive, not just short

Use typedef if it makes the code more readable

Condense code when you can do so without hurting readability
Debugging Exercise
Wrap Up
Reminders

Reading
   Read about Processes and Limited Direct Execution for Friday.

Assignment
   The ngram assignment is due next Monday at 11:59pm.
   Start early!

Need help?
   You can book a time during my office hours. See Teams or the class website.