Theory of Programming Languages

Some history and advice as we begin our adventure. . . .

Advice

- Attend every class. Don’t miss even one.
- Take notes in class, in pencil, on paper.
- Rewrite your class notes into a new notebook as a form of studying.
- Do every assignment and project.
- Work on your projects every day.
- Commit to GitHub frequently.

Advice

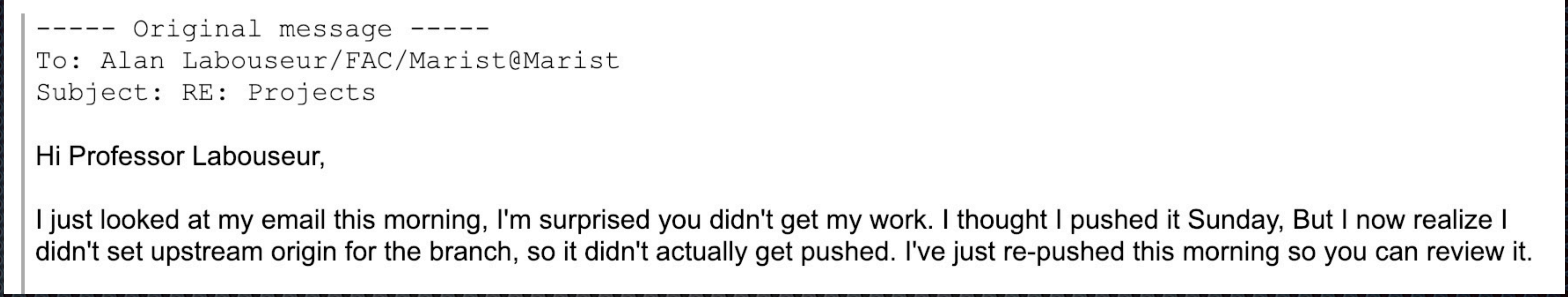
- Attend every class. Don’t miss even one. Why?
  - If you’re not in class I will never have the opportunity to invite you to join a faculty research project, or the IBM Joint Study, or a hackathon, or a CCAC project, or an ECRL project, or a Summer Fellowship, or the Marist Innovation Lab.
  - If you’re not in class I will never know you well enough to write you a letter of recommendation or recommend you to graduate school or an employer.

ADVICE from Dr. Helen Rothberg

- **A**ction — Do more, say less. Pursue your interests. Ask for help when you need it.
- **D**etermination — Stick to it and get things done.
- **V**ision — What are you trying to become? What do you want to do with your life? How will you get there?
- **I**ntegrity — Tell the truth all the time. Don't create or participate in drama.
- **C**ommunication — Be attentive and dig in, even if it's hard.
- **E**mpathy — Dare to care about yourself and others.

Do Not Make Excuses

- This is not acceptable:



----- Original message -----
To: Alan Labouseur/FAC/Marist@Marist
Subject: RE: Projects

Hi Professor Labouseur,

I just looked at my email this morning, I'm surprised you didn't get my work. I thought I pushed it Sunday, But I now realize I didn't set upstream origin for the branch, so it didn't actually get pushed. I've just re-pushed this morning so you can review it.

- I did not “review it”. The grade remained the same.
- If you cannot figure out GitHub, you do not deserve a Computer Science degree.

Academic Honesty

- This is college and we’re in an academic environment, where Academic honesty is paramount.
- You are not working on a personal side project or partaking in a hackathon, or in any situation where you might take shortcuts to get things done.
- This is college and **you must do the work** on your own.
- There are no shortcuts.

Academic Honesty

- These are individual projects.
All work must be your own.
- Any violation of this will result in your **immediate** and **automatic** failure of this class.
- This is your only warning. The next time we talk about this it will be in the context of me telling you that you have failed this class.
- I am serious. Ask prior students.

Write Original Code

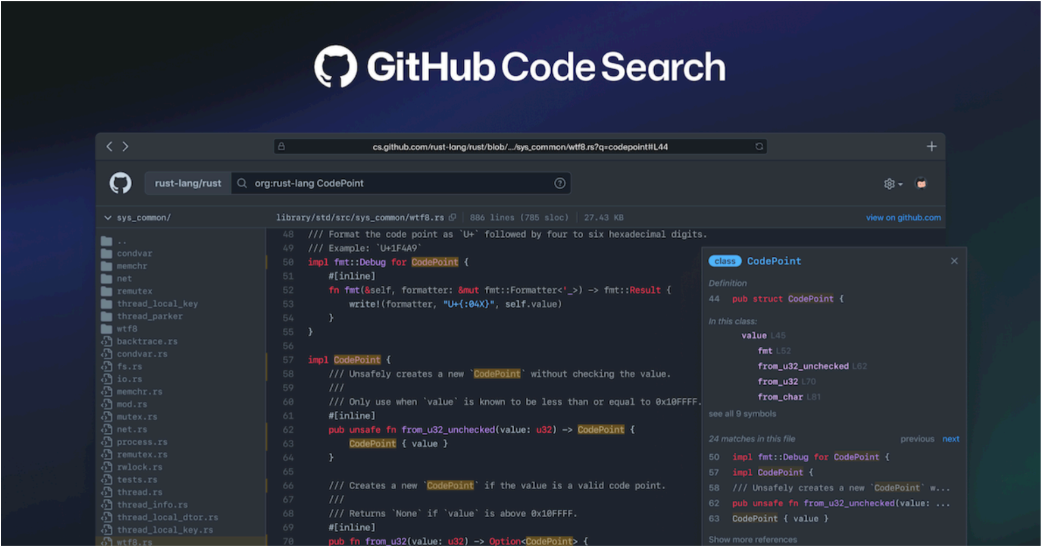
- Infuse your code with your personality.
  - variable names, code conventions, and style
  - structure and modularization

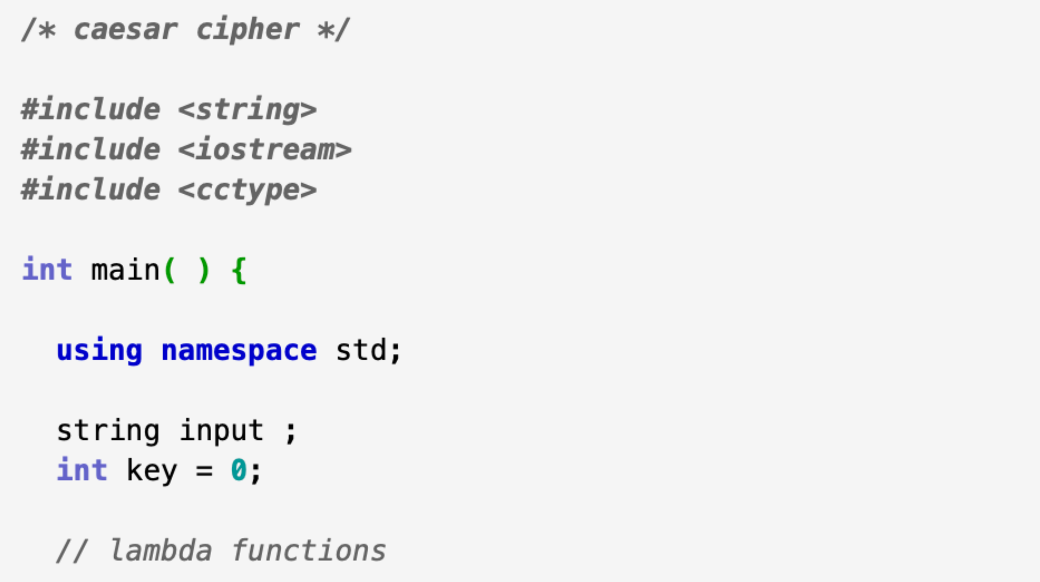
- Write grammatically correct comments and steep them with your personality. Be snarky, clever, and funny if you like. I want to get to know you by reading your code.
  - Comments should say things about your code that the code cannot say about itself.

- List collaborators, references, AIs, and sites consulted in a comment block at the top of each file.

Academic Honesty

- I will be checking your work against…
    - your class
    - other classes
    - past projects
    - GitHub
    - ~~GPTzero~~
    - and…
    - other Internet sources





```
/* caesar cipher */

#include <string>
#include <iostream>
#include <cctype>

int main( ) {

  using namespace std;

  string input ;
  int key = 0;

  // lambda functions
```

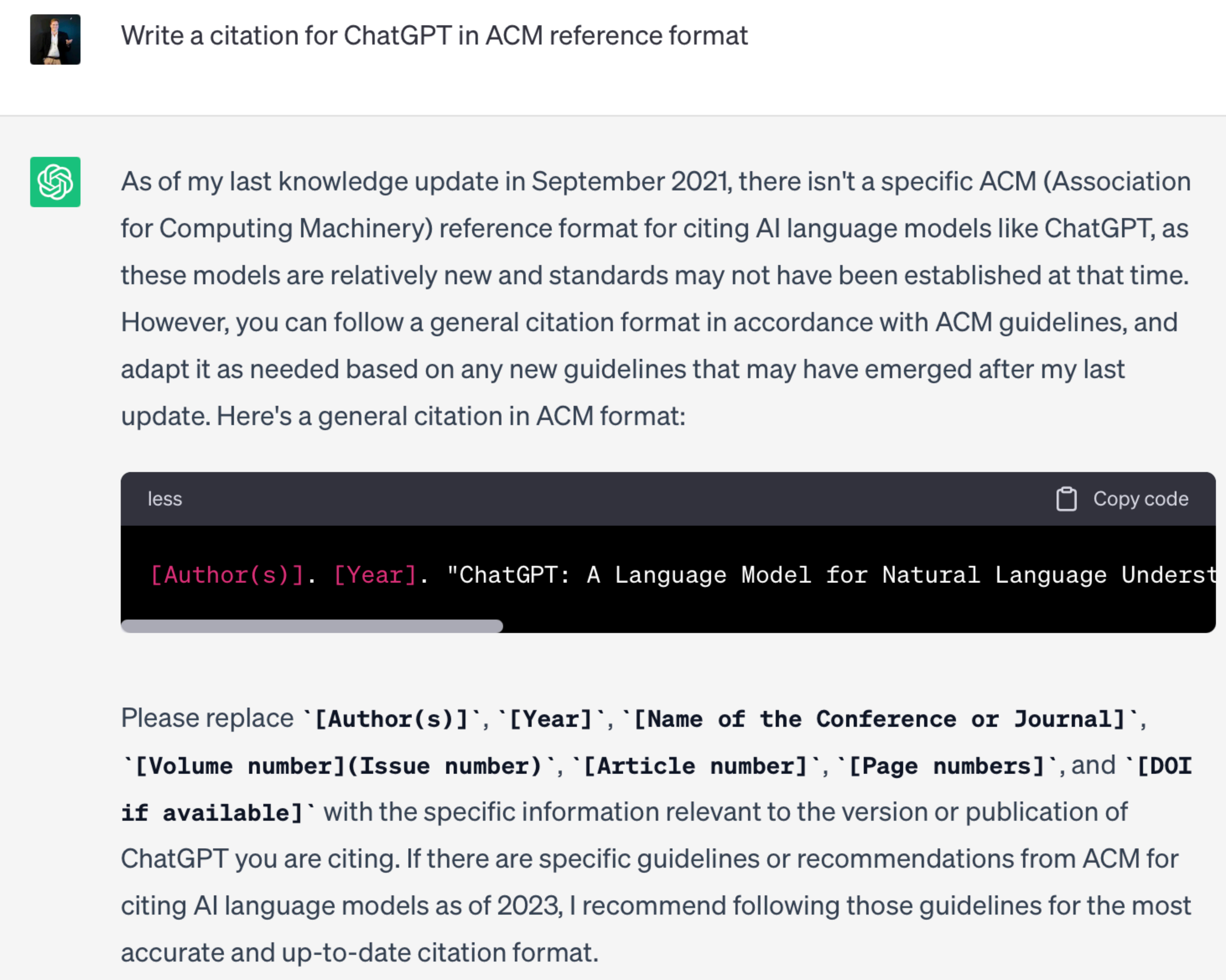
What about using an AI?

- I've been teaching my teenage son to drive. We started with golf carts and then moved to a real car in empty parking lots.
- He's making incremental progress built on a solid foundation (and cart paths). But he's a long way from driving in F1.
- If I were to put him in an Aston Martin he'd surely hurt himself and others.
- AI code generators are like F1 racecars. Until you have a solid foundation you're only going to hurt yourself and others by using them. But if you feel that you’re ready… DRS is enabled so go for it!

When using an AI

- As ever, anything that’s not 100% yours and yours alone **must be marked and cited with sources**. Whether you get help from an online AI, a tutor in the programming lab, or a random student sleeping outside of Marion Hall, you have to acknowledge that help and cite it or it is unquestionably plagiarism. Copying and pasting from a Hall of Fame project or GitHub or some other online source is unquestionably plagiarism unless you cite it; and this is no different.
- How do you cite it? Let’s ask ChatGPT . . .

When using an AI



Write a citation for ChatGPT in ACM reference format

As of my last knowledge update in September 2021, there isn't a specific ACM (Association for Computing Machinery) reference format for citing AI language models like ChatGPT, as these models are relatively new and standards may not have been established at that time. However, you can follow a general citation format in accordance with ACM guidelines, and adapt it as needed based on any new guidelines that may have emerged after my last update. Here's a general citation in ACM format:

```
[Author(s)]. [Year]. "ChatGPT: A Language Model for Natural Language Underst
```

Please replace `[Author(s)]`, `[Year]`, `[Name of the Conference or Journal]`, `[Volume number](Issue number)`, `[Article number]`, `[Page numbers]`, and `[DOI if available]` with the specific information relevant to the version or publication of ChatGPT you are citing. If there are specific guidelines or recommendations from ACM for citing AI language models as of 2023, I recommend following those guidelines for the most accurate and up-to-date citation format.

When using an AI

- I’m interested in learning about how well (or not?) AI-assisted software development works for you. If you make use of an AI you must document your experience with it for every project: the chat interactions, what worked and what didn’t, challenges, unexpected occurrences, etc. Include a reflection about the experience as well. I’m interested to learn from you and your experiences with this new technology.

- Remember: plagiarism is one of the few sins where it is easier to receive permission than forgiveness. So if you are at all unsure about any aspect of this, ask me about it first.

Before asking me for help

- Review your code yourself by reading it from the perspective of someone seeing it for the first time.
  - Consider your variable names — are they descriptive?
  - Did you write plentiful comments about why you made those choices rather than what the code does?
  - Examine your code conventions and style — is it consistent?
  - Fix anything that’s missing, unclear, or confusing.

When asking me for help

- Explain the problem clearly. Context matters.
- Describe several test cases that illustrate the problem.
- Tell me what have tried already.
- Make sure the latest version is committed on GitHub and include a link to it in your e-mail.