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 from Dr. Helen Rothberg

- **Action** — Do more, say less. Pursue your interests. Ask for help when you need it.
- **Determination** — Stick to it and get things done.
- **Vision** — What are you trying to become? What do you want to do with your life? How will you get there?
- **Integrity** — Tell the truth all the time. Don't create or participate in drama.
- **Communication** — Be attentive and dig in, even if it's hard.
- **Empathy** — Dare to care about yourself and others.
Do Not Make Excuses

This is not acceptable:

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

----- 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.
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.
  - comments
  - variable names
  - code conventions and style
  - structure
  - modularization

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