PPL 1/13

Reagan Shirk

January 13, 2020

Intro Stuff

- Don't use the last name Moreno even though OU lists it, he prefers Kong. Said we can call him Martin
- Office hours are DEH 230 from 11:00am-12:30pm
- Email is mkong@ou.edu
- Usually responds to emails in the morning between 8:00 and 8:30

Exam Schedule

- Midterm review: March 2
- Midterm: March 4
- Final: May 6, 4:30-6:30

Grades

- No attendance grade
- Quizzes: 30%
- Homework: 20%
- Midterm: 20%
- Final: 30%

Quizzes

- Online
- Beginning of class
- 15 minutes
- No makeup quizzes
- Open book
- Two lowest quizzes are dropped
- 8 total (5 before midterm, 3 after)
- Each is worth 5 points, no partial credit
- Quizzes are:
  - Jan 27
  - Feb 3
  - Feb 10
  - Feb 19
  - Feb 26
  - Apr 1
  - Apr 13
  - Apr 22

Homework

- mini-compiler project
- two before the midterm, two after the midterm
- given skeleton code to complete, grade is determined by the output
- need to know a bit of C
- uploaded to Canvas
- none of the homework is dropped
- 2-4 weeks for each homework
- each worth 5 points
- homework 1 is due Feb 5
- homework 2 is due Mar 2
- homework 3 is due Apr 1
- homework 4 is due Apr 29
- For late work:
  - -1 point for each day its late, up to 5 days
- If you use a Windows machine, you will need to install some Linux/Unix dependencies and such

What the course is about

- Fudnamentals of programming languages and compiler design
- How stuff works, can be a bit boring

Actual Material. . . woo

- Preliminaries
  - First language: machine instructions
    - With all computers, we had to code the instructions manually, basically in binary
  - 50s or 60s: Assembly was developed
    - Replace machine instructions with mnemonics so we don't have to remember stuff
    - macro expansion
  - Each computer used to have its own language (machine code)
  - Needed a way to reuse code, wanted the machine to be independent of language
  - 1950s came with the first version of Fortran followed my Lisp and Algol
  - Languages are implemented as compilers and interpreters
    - i.e. each language has a compiler or interpreter associated with it
    - compilers translate the "high level" shit we write to the machine instructions we don't wanna fuck with anymore
    - a fancy way to say what I said above: compilers translate the source language to the target language
      - The source language is usually described as a "high level" language, but it's all relative
    - compilers can also perform source-to-source translation and go from "low level" to "high level"
    - interpereters are similar to compilers but they execute commands
- Brief History
  - Language evolve/die by being adopted/neglected by different communities (say the machine learning community)
  - 60s and 70s: structured programming
    - Control flow (Fortran, Cobol, Basic) had explicit control with loops, if, switch, etc
  - 80s: Nested block structure
    - Algol, Pascal, Ada eas a precursor of object-oriented programming
  - 90s: Smalltalk, C++, Eiffel, Java

- 00s: scripting languages (PHP, Python, Ruby)
- Changes to programming languages come by different needs:
  - Productivity moved from machine instructions to assembly
    - Desire to write shorter, more compact code (array slices)
    - Resuability and maintainability (structures, functions, objects)
  - New areas and domains:
    - Classical scientific computations
    - Web design
    - Circuit design
    - Quantum computing