Fifteen students completed the anonymous background survey. Each standard statement has three numbers: the number who completely agree with it, the number who “sorta” agree with it, and the number who disagree with it. Thus 9 students were comfortable with fixed points, 2 were somewhat comfortable and 4 were uncomfortable. The statements are in descending order of universal agreement. Fill in the blanks responses are listed at the end — each student was given 1 total vote (so one student selecting C++ and Java would give both of them 0.5 votes).

- 15. 0. 0. I am comfortable with Turing machines and recursive functions.
- 15. 0. 0. I am comfortable with basic set theory, as in: \( \{ x \mid x \leq 5 \land x \in S \cup P(R) \} \).
- 12. 3. 0. I can learn new languages quickly.
- 12. 2. 1. I have taken a course that covered induction. I am comfortable proving things using induction.
- 10. 4. 1. I like homework assignments that also have implementation (programming) questions.
- 10. 4. 1. I like homework assignments that also have theoretical (logic) questions.
- 11. 1. 3. I am comfortable with context-free grammars (or Backus-Naur Form).
- 11. 1. 3. I do not mind if the instructor occasionally gets carried away and the class goes a bit over time.
- 9. 3. 3. I am comfortable reducing two-headed Turing machines to one-headed Turing machines, with or without a vorpal sword.
- 9. 2. 4. I am comfortable with fixed points: \( f(x) = x \).
- 9. 2. 4. I am comfortable with at least three of the following: Star Trek, Star Wars, Babylon Five, Battlestar Galactica, Doctor Who, The Twilight Zone, The X-Files, Futurama, Firefly, Farscape, or Buffy: The Vampire Slayer.
- 8. 3. 4. I am comfortable with Russell’s paradox, \( \{ A \mid A \notin A \} \), and notions like well-defined.
- 9. 0. 7. I have programmed in a commercial domain-specific language (e.g., QuakeC, UnrealScript, IDL, Infinity Engine Scripting, mIRC script, Matlab, or even HyperCard or Lua).
- 7. 4. 4. I am comfortable with the notions of completeness and consistency.
- 6. 3. 6. I am comfortable with a functional programming language (e.g., LISP, Scheme, ML, or even Python).
- 6. 2. 7. I have seen the lambda calculus: \( \lambda x.e, e_1, e_2, x \).
- 6. 0. 9. I have used an “automated” bug-finding tool (e.g., FindBugs, PREfast, JLint, PMD, Fortify, LCLint, Coverity, etc.).
- 6. 0. 9. I am familiar with at least two of the following incantations: wingardium leviosa, shoryuken, xyzzy, wonder twin powers: activate, for the honor of greyskull, spam spam spam humbug.
- 4. 4. 7. I can typeset documents in \( \LaTeX \).
- 5. 0. 10. I have seen typing judgments (rules of inference) like:  
\[
\begin{align*}
\Sigma & \vdash b : \text{bool} & \Sigma & \vdash e_1 : \tau & \Sigma & \vdash e_2 : \tau \\
\frac{}{\Sigma \vdash \text{if } b \text{ then } e_1 \text{ else } e_2 : \tau}
\end{align*}
\]
- 4. 2. 9. I have written an interpreter for a language with function calls.
- 4. 0. 10. I am comfortable with at least one of the following: Settlers of Catan, Carcassonne, Puerto Rico, Bohnanza, Lost Cities, Ticket To Ride, Diplomacy, or Niagara.
• 3. 2. 10. I can “prove” by induction that all horses are the same color.
• 2. 3. 10. I am comfortable with the syntax and semantics of the lambda calculus.
• 2. 2. 11. I understand and am comfortable with such typing judgments.
• 2. 2. 11. I am comfortable with the notion of truth with respect to a model (e.g., from a model theory course).
• 2. 1. 12. I have written a compiler that had a type checker.
• 1. 0. 14. I am comfortable with $\beta$-reduction and the Y-combinator.
• 1. 0. 14. I have used a formal verification tool or automated proof assistant (e.g., PVS, SPARK, NuPRL, Coq, or Spin).
• 0. 0. 15. I can name two Newbery Medal or Newbery Honor books, neither of which is Lois Lowry’s *The Giver*.

My favorite programming language is —
- C++ (3.5).
- Java (3.3).
- Scheme (3).
- Python (2).
- C (1).
- Perl (0.8).
- Pascal (0.3).

My least favorite programming language is —
- Perl (4).
- Fortran (3).
- C++ (1.5).
- C (1).
- nesC (1).
- OCaml (1).
- Python (1).
- R (1).
- VB (1).
- Ada (0.5).

My favorite non-PL area of CS is —
- Security (3.3).
- Algorithms / Theory (3).
- Architecture (2).
- Software Engineering (1.3).
- Artificial Intelligence (1).
- Bioinformatics (1).
- Debugging (1).
- Simulation (1).
- Data Mining (0.3).

I would like to learn from this course —
- How PL relates to security (2).
- Type systems and theory (2).
- Evaluate new language designs (2).
- Get the basics of PL (2).
- New languages.
- Symbolic execution.
- Abstract interpretation.
- Theorem proving.
- Find a research topic.
- Understand the CQual (type qualifiers) paper.
- Help with quals.
- “Advanced topics”.

3:30-4:30 is a good time for Weimer’s Wednesday Office Hours. Only (2) dislike this time.