| **Instructor:** | Professor Tom Shrimpton  
FAB 115-13  
email: teshrim@cs.pdx.edu  
Phone: 725-4055 |
| **TA:** Caylee Hogg | TBA  
email: TBA |
| **Office Hours:** | Tom: M 12:30-1:30pm and by appointment  
TA: TBA |
| **Required Textbook:** | *Introduction to the Theory of Computation, 3rd Ed.*  
Michael Sipser (2nd Ed. is just fine) |
| **Recommended Textbook:** | *Computational Complexity: A Modern Approach*  
Sanjeev Arora and Boaz Barak |
| **Prerequisites:** | CS 311 (or equivalent) |
| **Grade and Grading Policy:** | Homework 40%, Midterm Exam 30%, Final exam 30%. I reserve the right to make slight adjustments (up or down) to your grade to reflect your participation and effort in the course. The grading scale will be the standard one: 90-100=A range, 80-89=B range, etc., but the mapping from “raw” overall course percentage to final overall course percentage won’t be determined until the end of the course. (I do give ± modifiers to letter grades, and each range includes these.)  
I tend to grade strictly and I don’t give partial credit just because you’ve written a lot. This is for your benefit, as you’ll have a realistic view of your understanding and progress. If you consistently put in solid effort, I’ll notice, and you’ll be rewarded! (Likewise if you consistently turn in shoddy work, although with the opposite outcome.) |
| **Homework Policy:** | You may collaborate with other class members on the homework assignments. If you do so, please turn in one solution set for the group. No more than three students in a group, please! Solutions must be neat and clear, preferably typeset in \LaTeX. |
| **Important Dates:** | First Exam: late Jan; Second Exam: late Feb; Final Exam: Monday, March 17 (but may be take-home) |
Potential Course Topics

Block 1: Languages and Computability
1. Regular langs, DFAs, NFAs, REs, RE→(minimal) DFA ......................... (Sipser 1)
2. Context-free langs, CFGs, basic parsing algos ........................................ (Sipser 2)
3. Turing Machines and variants ................................................................. (Sipser 3)
4. (Un)Decidable Languages, mapping reductions, Rice’s theorem ............. (Sipser 4,5)

Block 2: Time Complexity
5. The classes P, NP, NP-completeness ...................................................... (Sipser 7)
6. Poly-time reductions, lots of NP problems, Cook-Levin theorem ............ (Sipser 7)

Block 3: Space Complexity
7. The classes PSPACE, NPSPACE, PSPACE-completeness ..................... (Sipser 8)
8. Savitch’s theorem ..................................................................................... (Sipser 8)

Block 4: Circuit Families and Non-uniform Computation
9. Circuit families, circuit complexity, uniform vs. non-uniform computation ... (Sipser 9.3 +)
10. The class P/poly and its relationships to other classes ......................... (+)

Block 5: Randomized computation and Approximation:
11. Randomized TMs, and the classes BPP, RP, ZPP .................................... (Sipser 10.2 +)
12. Entropy, computational entropy, and randomness extraction ................. (+)
13. Pseudorandomness (“randomness” from computational hardness) .......... (+)
14. Basics of approximation algorithms, hardness of approximation .......... (Sipser 10.1 +)
15. Worst-case vs. Average-case complexity (?) ........................................... (+)
16. Probabilistically checkable proofs and the PCP theorem (?) .................... (+)

Block 6: Interaction and Proof Systems:
18. Interactive proof systems and the class IP ............................................. (Sipser 10.4 +)
20. Zero-knowledge proofs ............................................................................ (+)
21. IP = PSPACE, and a proof sketch(?) ................................................... (Sipser 10.4)

Block 7: Quantum Computation
22. Quantum computing models ................................................................. (+)
23. Quantum algorithms, e.g. Shor’s ......................................................... (+)

I don’t know exactly which topics we’ll manage to cover this term, since this is my first time teaching CS 581, and there are so many interesting topics! For certain, we’ll cover Blocks 1-3.

The Sipser text does a nice job of covering the material that it covers, but much of Blocks 5-7 are not really covered by it. For those topics, should we take them on, we’ll need to use other sources. (Whenever I think we’ll need to go outside of Sipser, I’ve denoted the topic with a +.) The Arora and Barak text is nice, if a bit advanced. I might also draw from some classic papers.
Questions You Might Ask

• **Will there be any programming assignments?**  Nope.

• **Does that mean the course will be easy?**  Nope.

• **What makes this course hard?**  Three things: formalism, abstraction, and proofs. Building up a precise, abstract model of a real-world thing (like an elevator controller, a programming language, a computer, a network protocol) is hard! Separating what a thing is from what a thing does takes careful thinking, and being formal and precise is a lot harder than waving your hands and saying “well, you know what I mean.” Creating proofs —real, convincing, mathematically meaningful arguments— requires an appetite for frustration and (temporary) defeat that takes time to develop.

• **I didn’t do well in my undergraduate automata/complexity course (e.g. CS 311). Will I be okay in 581?**  Almost certainly not. Moreover, we’re not going to spend the course re-doing CS 311. If you’re not prepared, be honest with yourself, and talk to me and/or the graduate program coordinator about alternative plans.

• **Why should I care about this useless theory stuff?**  First of all, it’s good for you. Classes like this are what change you from programmers into Computer Scientists. Secondly, it’s interesting! We’ll be exploring fundamental ideas of what can(not) be computed, and what can(not) be computed efficiently. We’ll think about what it means for a problem to be efficient, formally speaking, both in terms of running time and memory usage. We’ll explore problems that are efficiently solved if you allow the computing device to flip coins, or to have pre-computed hints. We’ll see that some computational problems are believed to be so hard that even getting a close approximation of the correct answer is hard. We’ll look at models of two-party protocols, in which one (the prover) tries to convince the other (the verifier) that some statement is true; potentially, without the verifier learning anything about the statement, other than its truth!

Oh, also, these ideas and techniques come up in cryptography, machine learning, network algorithms, algorithms on data streams, . . .

• **Any hints on doing well in 581?**  (1) Don’t fall behind, because the material in the class really does build on itself. (2) Put aside some time every day, even when there isn’t a pending assignment, to think about the material. Re-read a previous section, work an extra problem, clean up your notes, just do something 581-related. It’s amazing how effective that is. (3) Don’t try to fool me, or yourself, by slapping together an answer that you think probably isn’t right... but maybe you’ll get lucky, and maybe I won’t notice. You won’t; I will. (4) Don’t cheat. And, yes, using someone else’s solutions that you found on the web is cheating. (It is almost always obvious when this happens.) If you find yourself thinking about cheating, come to me or the TA for help. I’d much rather help you to learn than fail you for cheating.