This course provides a challenging introduction to some of the central ideas of theoretical computer science. It attempts to present a vision of “computer science beyond computers”: that is, CS as a set of mathematical tools for understanding complex systems such as universes and minds. Beginning in antiquity—with Euclid’s algorithm and other ancient examples of computational thinking—the course will progress rapidly through finite automata, Turing machines and computability, decision trees and other concrete computational models, efficient algorithms and reducibility, NP-completeness, the P versus NP problem, the power of randomness, cryptography and one-way functions, computational learning theory, interactive proofs, and quantum computing and the physical limits of computation. Class participation is important, as the class will include discussion and debate about many of these topics.

Requirements. Students taking 6.045 will be graded on the following basis:

- 40% psets
- 20% midterm exam
- 30% final exam
- 10% class participation

Textbook. There is no required textbook for the course. However, there are three recommended books:

2. *Introduction to the Theory of Computation* by Michael Sipser – covers most material from the first half of the course


**Psets.** There will be 6-7 psets, which will generally be due a week and a half after being assigned. Late psets will receive half credit, except in special circumstances such as illness or family emergency. Psets can be turned in to the TAs (in class, recitations, or office hours) or submitted via Stellar; we will also look into setting up “pset bins.” Pset problems will often ask you to prove theorems, but we’re looking for clarity of understanding rather than “rigor for rigor’s sake.” So for example, a brief verbal description of an algorithm is better than complicated spaghetti code; and a clear explanation of how you tried to solve a problem and failed will earn partial credit, while obvious nonsense won’t!

**Collaboration Policy.** You are welcome to collaborate on psets, provided that (1) you write up your solutions individually, and (2) you list the names of all collaborators.

**Prerequisites.** We assume that you have taken 6.042 Mathematics for Computer Science, or have equivalent mathematical preparation. In particular, we will assume you have basic “mathematical maturity”: i.e., that you are comfortable both reading and writing mathematical proofs.

**Syllabus (extremely approximate).**

**Logic, Math, and Machines**
- Ancient computational thinking (Euclid et al.)
- Finite automata
- Turing machines and the halting problem
- Oracles and computability
- Gödel’s completeness and incompleteness theorems
- Philosophical considerations (Penrose and “strong AI”)

**Computational Complexity**
- Decision trees and circuits
- Polynomial time and its justification
- Nontrivial examples of polynomial-time algorithms
- The concept of a reduction
- P, NP, and NP-completeness; the Cook-Levin Theorem
- The P versus NP problem and why it’s hard

**Randomness, Adversaries, and the Physical World**
- The power of probabilistic algorithms
- Private-key cryptography and one-way functions
- Public-key cryptography and trapdoor functions
- Pseudorandom number generators
- Does randomness really help? The P versus BPP question
- Zero-knowledge proofs
- Computational learning theory
- Quantum computing
- The ultimate physical limits of computation