Computability

People have been making calculations as long as there have been numbers. The question: "What can be calculated?" is almost as old. The Greeks wondered about "constructable" numbers that represented lengths of line segments that could be constructed with a compass and straightedge.

The question "What can be calculated?" became more formal, and more answerable, in the 19th Century. This question became entwined with questions about the foundations of mathematics (because before that it was assumed that *everything* could be calculated).

In 1901 Bertrand Russell discovered his paradox: "Let S be the set of all sets that don’t contain themselves. Is S a member of S?" The answer is obviously neither "yes" nor "no". Can one direction or the other be proven? If so, logic is useless; we can prove things that aren't true. If not, Russell's paradox is a statement that can be neither proven nor disproven.

Now, provability and calculatability aren't quite the same thing, but they are clearly related.

From the late 19th Century into the 1930s Combinatory Logic was developed by Moses Schonfinkel and others to explain what could be calculated or proven. Schonfinkel based Combinatory Logic on functions rather than predicates and developed the idea of a recursive function long before there were computers and programming languages.

The most famous mathematician in the early 20th Century was David Hilbert, a professor at the University of Gottigen in Germany. Hilbert was a particularly famous poser of questions; anyone who answered on of his questions became instantly world-famous. In 1928 he pronounced the "Entscheidungsproblem" (Decision Problem): Could every question with a yes/no answer be decided algorithmically? In 1928 Hilbert, and everyone else, thought the answer was obviously "yes" but didn't know how to prove it.

In 1931 Kurt Godel proved his Incompleteness Theorem, which showed that in any mathematical system rich enough to include basic arithmetic, propositions could be expressed that could neither be proved not disproved.

This did not directly address the Entscheidungsproblem, but after 1931 everyone guessed that the answer to the Entscheidungsproblelm" was "no".

In 1936 Alonzo Church (at Princeton) invented the lambda calculus to show that the answer to the Entscheidungsproblem was "no". Later that same year Alan Turing invented Turing Machines to show again that the answer to the Entscheidungsproblem was "no". This set limits on the notion of what could be calculated.

After WWII, as interest in computation grew, there were numerous models of computability including

- Turing Machines
- Church’s lambda-calculus
- Recursive Function theory
- Emil Post's "correspondences"

These were all shown to be equivalent. Church’s Thesis (first formally stated by Stephen Kleene in 1943) says that Turing Machines (hence all of the others) embody our informal notion of what it means to be an algorithm

In this class we will look at a number of different mathematical models of "computability". These all have practical ramifications for anyone who write programs and we will look at those implications, but our main focus will be on the philosophical question "What problems can be solved by a computer?"