Goal of this Class

Teach you how to prove mathematical theorems formally, such that a computer can check your work.

We'll be using the Coq proof assistant exclusively.

I want the class to be useful to people interested in theorem proving in general, though my research focus is in program verification (hence the title of the course textbook).
My Unusual Style

The style of Coq usage that I follow in this class is very different from what almost anyone else is using.

...but I hope you (eventually) agree that it's much more productive.
Class Web Site

http://stellar.mit.edu/S/course/6/fall/6.892/
Course Staff

Me: Adam Chlipala [http://adam.chlipala.net/]
Fresh assistant professor in CS
Looking for research students!
Research applies this stuff to program verification, programming language design, etc.
No TA's, etc.
I think the material is new to everyone else at MIT. ;)}
Prerequisites

Basic discrete math and logic
Some programming experience
[more on this later]

Not needed: Any experience with formal semantics of programming languages or mechanical theorem proving.
Survey

Who has already used a computer proof assistant?

Who already feels comfortable with Haskell, ML, or another typed functional programming language?
Requirements for Taking the Class for Credit

Coq problem sets (30% of grade)
100% computer-checked assignments; the only way not to realize you made a mistake is to formalize theorem statements incorrectly.

Research project (50% of grade)
Use Coq to prove some significant theorem.

Class participation (20% of grade)
For non-MIT people

Everyone is welcome in the class, taking it for credit or not!

E-mail me if you'd like to be granted full access to the course web site.

(I wish I could make it all public, but I haven't found a setting for that.)
How I Hope “Lectures” Will Go

We walk through code examples in a live IDE.

The class figures out together how to prove the theorems.

Digressions encouraged, based on what-if questions from the audience.

Hopefully this is the last lecture with bullet points on slides. ;-}
Texts

My own textbook, which I've been working on since 2008

Available free online [http://adam.chlipala.net/cpdt/]

Linked from course home page

The Coq reference manual, at
http://coq.inria.fr/

The “Coq'Art book” might be useful, but shouldn't be necessary.

Google “Coq'Art” to find it.
The Book & Lectures

The book is generated from Coq source files, so you can step through it interactively.

Lecture example code will come out of the book.
Course Map

**Today:**
Administrivia
Demo to preview the fancy stuff we'll learn how to do

**Next class:**
Tutorial on typed functional programming
mostly old hat to Haskell/ML programmers, but will introduce how usual things are done

---

**Class**

- Roughly 1/3 of lectures:
  Core material for productive Coq hacking

**Other**

- Problem sets
- Think of project ideas
- More advanced material
- Researchy projects: prove something interesting in Coq

- You demoing your projects
Next Class: Typed Functional Programming Tutorial

Introducing Coq versions of the core shared between ML, Haskell, and many other functional languages

The class will be in a sort of “lab” format, so bring a laptop! (Or find a partner who will)

Get Coq installed before class

http://coq.inria.fr/
Projects

Main part of the class experience will be a research-y project that involves proving an interesting set of theorems in Coq.

Teams are OK, though project complexity should be commensurate with team size.

Start thinking of ideas and be ready to do an official proposal by about the 1/3 point of the semester.
Two Kinds of Projects [not exhaustive]

Complement your research
Formalize some aspect of your research and prove something about it.

Toolbox for a class
Pick a class or textbook that you like.
Pick some well-defined kind of exercise in it.
Build a Coq library that makes it practical to solve that kind of exercise with formal proofs, with reasonable human effort.
I'm Available to Help!

I'm willing to devote a lot of time to helping you with your projects.

[I'm brand new, without students and looking for collaboration opportunities. :-)]

Will have at least one check-in point where you give an update on progress.

Best to choose projects with several goals of increasing ambitiousness, so you can scale as time permits.
Machine-checked mathematical proofs?
The Four Color Theorem

[Appel & Harken, Georges Gonthier]

Proof identifies an “unavoidable” set of 1476 “reducible” maps

Check entire argument in Coq, with no need to trust in proof details

Manual auditing

Check with software
CompCert Verified C Compiler
[Xavier Leroy]
Proof Assistant Comparison

General-Purpose PL
Higher-Order
Polymorphism
Dependent Types
Small Proof Language

* Includes Isabelle/HOL and others