Course Projects

The final project is due at the beginning of lecture on Wednesday, May 14, the last day of class. Possible projects include:

1. Any course-related project you believe will be valuable for yourself and/or will enrich the course materials for use in future terms. Lecturer approval required.
2. Extend the submodel implementation in some interesting and/or useful way, eg, by implementing the revised call/cc rules in Notes 3, section 11.5.
3. Verify that rewriting still preserves observational equivalence using the revised rules for call/cc in Notes 3, section 11.5.
4. Carefully work out a solution to Notes 3, Problem 16 – be careful, the claimed result may be buggy.
5. Computability, Part I (Notes 5), Prob.22: generalize Scott's Rice's 2nd Theorem.
6. Work out the 2nd Incompleteness Theorem proof outlined in Computability, Part II (Notes 6), Section 6.
7. Given expressions, *F*,*G*, respectively computing functions, *f*,*g*, on S-expressions, Lemma 3.4 in Notes 5 observes that the expression

   ```
   (lambda (s) (F (G s)))
   ```

   computes *f o g*, the composition of *f* and *g*. The proof in the Notes of this claim is not very satisfactory because it rests on a variety of reasoning principles that were not fully spelled out or justified, for example, rules like

   ```
   F converges to (letrec (B) V),
   (letrec (B) M) converges to (letrec (B') V'),
   ----------------------------------------------------------------
   (F M) converges to (letrec (B') V').
   ```

   Use properties of the Substitution Model to verify this rule and any similar rules needed to develop a thorough proof of Lemma 3.4.

   Then go on and try to develop a *complete* proof system for assertions of the form "*M* converges to *V*" where *M* is an expression and *V* is a final value in Scheme without `call/cc`.

8. Find an explicit arithmetic inequality that can't be proved using the final proof system for inequalities in Notes 1 on *Proving Arithmetic Equations*, and prove that it can't be proved.
9. **Notes 3, Problem 13:** carefully prove that rewriting preserves observational equivalence. (This was also listed as a possible PS2 problem, but done properly merits Project status. As a project, it was initially phrased here as "Carefully

prove Corollary 10.3 of Notes 3 on the Substitution Model." Corollary 10.3 is labelled Lemma 9.4 in the latest revision of Notes 3.)

10. Identify some other poorly written or inadequately explained portion of the course Notes or Scheme Code, and improve it. Lecturer approval required.