Homework 9: Denotational Semantics

Due: May 4 (no extensions possible, as the final is May 9)
This homework is to be done individually.
This homework is about denotational semantics.
For all SML programs, provide evidence that your program is correct (for example, test cases).
Hand in a printout of your code and your testing. You should program in a purely functional style (for example, don't use `ref`).
Many of the following problems are taken from Watt's book *Programming Languages Syntax and Semantics* (hereinafter referred to as "Watt's book"). Note that there are answers to some of Watt's problems in appendix A; these may be of some help to you if you wish to try other exercises and check your answers to those. I've suggested a few especially good problems of this type; don't hand these in, but think about them and check your answer as a study aid.

1. (10 points) Do problem 3.10 on page 94 of Watt's book *Programming Languages Syntax and Semantics* (adding booleans).
2. (0 points, practice then look at the answer; applications, semantics) Try problem 3.7 on page 93, and then look at the answer in appendix A.
3. (0 points, practice then look at the answer; semantics) Try problems 3.11 and 3.12 on page 94, and 3.15 on page 95, and then look at the answer in appendix A.
4. (15 points) Do problem 3.20 on page 96 of Watt's book (value-result parameters).
5. (10 points) Do problem 4.1 on page 118 of Watt's book (reading semantic equations).
6. (0 points, practice then look at the answer; semantics) Try problems 4.2 and 4.4 on page 118, and 4.8 and 4.9 on page 119, and then look at the answer in appendix A.
7. (28 points) Do problem 4.5 on page 119 of Watt's book (reading a denotational semantics).
8. (30 points) Read Watt's section 4.4 and then modify the semantic prototype of mini-Cecil found in the directory

   ```
   ~leavens/src/sml/mini_cecil
   ```

   in the same way as in the previous homework. That is, your tasks are to: (a) modify the prototype so that one cannot assign to the formal parameters of a method, and (b) add syntax and semantics for closure expressions. Your modified semantics should be able to work with curried procedures! Hand in any code you modified, along with test cases.
9. (60 points, extra credit only; applications—databases, semantics) Give a denotational semantics to a subset of some database query language.
10. (60 points, extra credit only; semantics) Make the semantics of mini-Cecil closer to Cecil, by adding subtyping, and multi-method dispatching. To do this, each object should keep track of its class, and method lookup should choose the unique most specific method, if it exists, and give an error otherwise.
11. (15 points) Read chapter 9 of Watt's book. Imagine that you are working for a company that is designing a new spreadsheet, wordprocessor, or database system. Answer the following questions. Is it likely that anything you learned about operational or denotational semantics will be of use in such a project? Why or why not?