CSE 341, Autumn 2008, Assignment 5
Scheme Macros
Due: Friday Nov 7, 10:00pm

8 points total (4 points each question)

You can use up to 2 late days for this assignment.

1. The lecture notes and code for `delay` and `force` included functions `my-delay` and `my-force`. Rewrite `my-delay` as a macro, so that the user doesn't have to manually wrap the delayed expression in a lambda. (So the syntax for `my-delay` should be just like `delay`.) Rewrite `my-force` so that it works correctly with expressions that weren't delayed using `my-delay` (just like `force`). Leave `my-force` as a function though; you don't need to make it a macro.

2. Define a macro `my-and` that does exactly the same thing as the built-in Scheme special form `and`. (Hint: see the handouts for macros, in particular the `my-or` example. Remember that `and` works on an indefinite number of expressions, including 0 expressions.)

Turnin: Turn in your Scheme program and a script showing it running on some well-chosen test cases. For delay and force, show that the delayed expression is not evaluated until it is forced, and that it is evaluated only once, even if forced several times. Also show that force applied to an object other than a delay just returns that object.

As usual, your program should be tastefully commented (i.e. put in a comment before each function definition or macro saying what it does).