Recitation 19: Michael Collins

(1) Say that we evaluate the following statement in the evaluator:

\[
\text{m-eval '}(\text{define x 6) the-global-environment})
\]

what is the return value in this case?

How would you alter the code so that define statements of the form \(\text{define name exp}\) always returned the value of the \(\text{exp}\), rather than some undefined value?

(2) Create code to process the special forms \textbf{and} by extending the \textbf{cond} in \textbf{m-eval} and writing the procedure \textbf{eval-and}. 
(3) We’ll now create code that adds and to the evaluator, using syntactic sugar. You should write the code that converts an and statement to a statement involving if, then passes that to the evaluator.

First, say we have the statement

\[ \text{and} (> x 4) (< y 5) (> z 6) \]

What would be an equivalent if statement?
(Hint: you can use an if statement combined with an and statement that only has two clauses.)

Now write the code and->if that performs this conversion, for example
\( \text{and->if '}(\text{and} (> x 4) (< y 5) (> z 6)) \) should produce the correct if statement. Note that your code should handle the expression (and), which evaluates to #t.

How would you define eval-and to make use of and->if?