CSE 341 — Racket Discussion Questions Part 3

These questions deal with macros, delayed evaluation, improper lists, and functions with a variable number of arguments.

1. The lecture notes for macros include a definition for \texttt{my-or} that works just like the built-in \texttt{or} in Racket.

   \begin{verbatim}
   (define-syntax my-or
     (syntax-rules ()
       ((my-or) #f)
       ((my-or e1 e2 ...) (let ([temp e1])
                          (if temp temp (my-or e2 ...))))))
   \end{verbatim}

   Given this definition, if we expand \texttt{(my-or (= x 2))} we get

   \begin{verbatim}
   (let ([temp (= x 2)]) (if temp temp (my-or)))
   \end{verbatim}

   This would further expand to

   \begin{verbatim}
   (let ([temp (= x 2)]) (if temp temp #f))
   \end{verbatim}

   Modify the rule so it just expands \texttt{(my-or (= x 2))} to \texttt{ (= x 2)} instead. It should still work correctly for \texttt{(my-or)}.

2. Suppose we are writing our own version of the \texttt{if} special form, called \texttt{my-if}. This can’t be a normal function in Racket, since we evaluate the arguments. We can write it as a macro, of course. For this mini-exercise, write it as a function that uses delay to delay evaluating some or all of the arguments. Only delay arguments if need be.

   Now rewrite the following expression using your \texttt{my-if} function. (All you need to do is insert the appropriate delays. Use Racket’s built-in delay macro.)

   \begin{verbatim}
   (if (= 1 1) (+ 2 4) (/ 10 0))
   \end{verbatim}

3. Draw a box-and-arrow picture for the value of \texttt{'(squid . (clam . (octopus . ()))))}

4. Draw a box-and-arrow picture for the value of \texttt{'(squid . (clam . octopus))}

5. How would you write the following list structure in Scheme?

   \begin{verbatim}
   _____________ _____________
   | | | | | / |
   o ----|-----> o | |
   | ___|___|_______| |___|___|/______|
   | |
   | |
   | |
   ___|___________ ______|___________
   | | | |
   1 | 2 | 3 | 4 |
   ___|_______|_______|
   \end{verbatim}
6. Write a function \texttt{my-max} that finds the maximum of its arguments. It needs at least one argument, and can take arbitrarily more. For example

\begin{verbatim}
(my-max 4 10 2 1)
\end{verbatim}

should return 10.

You can use a helper function if you need to. Bonus points though for a version without a helper function!

What do these evaluate to?

\begin{verbatim}
(my-max 3)
(my-max)
\end{verbatim}