ATOMS, PROCEDURES, EXPRESSIONS, AND EVALUATION.

1. In Scheme, the smallest unit is called an atom. Atoms may be a: number, symbol, boolean, string, or character.

For each of the following, indicate which of these five types it is. Write your answer next to each atom, or “error” if it’s syntactically incorrect.

3                  'true
17.2               "true"
3/4                #t
-1                 0x41
'foo               #\A

2. Procedures (or functions) are entities that manipulate data. When you initialize a Scheme environment, a lot of built-in procedures are automatically loaded and ready to use. E.g.:

+ is a procedure for adding numbers (which evaluates to a number)
abs is a procedure for computing the absolute value of a number (which evaluates to a number)
= is a procedure for determining whether numbers are equal (which evaluates to a boolean)
string? is a procedure for determining whether its argument is a string (which evaluates to a boolean)

Question: Do atoms come with information about which type they are? Circle yes or no.

3. Procedures may be applied to their parameters (or arguments) in an expression. This is done by writing the procedure followed by its parameters in parentheses.

This process of procedure application is called expression evaluation (or simply, evaluation).

For each of the following expressions, write next to it the value that will result from its evaluation. Or, write “error” if you think it’s an invalid expression.

(+ 3 4)          (+ 3 4 5)
(+ 3 (* 4 5))            (+ 3)
(+)                        (+ 3 'foo)
+

4. User procedures may be created with the special form \texttt{lambda}. For example, the following expression creates a procedure that will mathematically square its parameters:

\texttt{(lambda (n) (* n n))}

Note the form of this expression: the keyword \texttt{lambda}, a list of its parameters, and an expression which becomes the procedure’s body.

Questions:

How many parameters does the procedure have?

What is the result of evaluating the expression above?

What is the result of evaluating this expression?

\texttt{((lambda (n) (* n n)) 3)}

5. New \textit{bindings} are created in an \textit{environment} with the special form \texttt{define}. E.g.:

\texttt{(define foo 3)} \hspace{1em} ; creates a binding from a symbol named \texttt{foo} to the value \texttt{3}
\texttt{foo} \hspace{1em} \hspace{1em} \hspace{1em} ; causes \texttt{foo} to be evaluated. This lookup yields a \texttt{3}
\texttt{(define square (lambda (n) (* n n)))} \hspace{1em} \hspace{1em}; binds \texttt{square} to a procedure object
\texttt{(square 5)} \hspace{1em} \hspace{1em} \hspace{1em} ; \texttt{25}

Questions: What will evaluating the following yield (or “error”)? Assume the prior bindings still exist.

\texttt{(square foo)}

\texttt{(square 'foo)}

\texttt{(foo square)}