PROCEDURES, EVALUATION, IF, AND/OR

1. 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:

\begin{verbatim}
(lambda (n) (* n n))
\end{verbatim}

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?

\begin{verbatim}
((lambda (n) (* n n)) 3)
\end{verbatim}

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

\begin{verbatim}
(define 'foo 3) ; creates a binding from a symbol named \texttt{foo} to the value 3
foo ; causes \texttt{foo} to be evaluated. This lookup yields a 3
(+ foo 7) ; that’s 10
(define square (lambda (n) (* n n))) ; binds \texttt{square} to a procedure object
(square 5) ; 25
\end{verbatim}

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

\begin{verbatim}
(square foo)
(square 'foo)
(foo square)
\end{verbatim}
3. The `if` expression has the form:

```
(if test-expr then-expr else-expr)
```

It evaluates `test-expr`. If it produces any value other than `#f`, then `then-expr` is evaluated, and its results are the result for the `if` form. Otherwise, `else-expr` is evaluated, and its results are the result for the `if` form¹. This is a special form.

Write the values that will result from evaluating the following expressions, or “error.” Assume the environment is cleared before the beginning of each line.

```
(if (= 1 1) 3 4)
(if 'blah 'foo 'bar)
(define foo 3) (if #t foo bar)
(define foo 3) (if #f foo bar)
```

4. `and` and `or` are special forms. Each takes an arbitrary number of parameters (none is OK too), and evaluates its parameters from left to right, until the result is known. Then it stops evaluating and results the result (short-circuiting).

Write the values that will result from evaluating the following expressions, or “error.”

```
(and #t #t)               (or #t #f)
(and #t #t #f)             (or #t foo)
(and #f foo)              (or #f foo)
(and)                     (or)
```

¹  https://docs.racket-lang.org/reference/if.html