<table>
<thead>
<tr>
<th>Question</th>
<th>Score (from)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 (from 20)</td>
</tr>
<tr>
<td>2</td>
<td>0 (from 10)</td>
</tr>
<tr>
<td>3</td>
<td>0 (from 10)</td>
</tr>
<tr>
<td>4</td>
<td>0 (from 20)</td>
</tr>
<tr>
<td>5</td>
<td>0 (from 10)</td>
</tr>
<tr>
<td>6</td>
<td>0 (from 15)</td>
</tr>
<tr>
<td>7</td>
<td>0 (from 15)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>0 (from 100)</strong></td>
</tr>
</tbody>
</table>
Write the function \( \text{max0} : \text{list-of-numbers} \rightarrow \text{number} \). It computes the maximum number in a list of numbers, assuming that all of the numbers are positive.

**Solution**

\[
\begin{align*}
(\text{define} \quad (\text{max0} \; \text{lon}) \\
(\text{cond} \\
\quad [(\text{empty?} \; \text{lon}) \; 0] \\
\quad \text{else} \; (\text{max-num} \; (\text{first} \; \text{lon}) \; (\text{max0} \; (\text{rest} \; \text{lon})))))) \\
\end{align*}
\]

;;;; \text{max-num} : \text{number} \; \text{number} \rightarrow \text{number} 

\(\begin{align*}
(\text{define} \quad (\text{max-num} \; a \; b) \\
(\text{cond} \\
\quad [(< \; a \; b) \; b] \\
\quad \text{else} \; a))) \\
\end{align*}\)

Or, if you looked ahead and are into re-use, you might have written:

\[
\begin{align*}
(\text{define} \quad (\text{max0} \; \text{lon}) \\
(\text{cond} \\
\quad [(\text{empty?} \; \text{lon}) \; 0] \\
\quad \text{else} \; (\text{max} \; \text{lon})))) \\
\end{align*}
\]
Ferns have a funny, replicated kind of structure. Look at the picture below. Each of the parts sticking out to the side looks a lot like the whole fern, but smaller and pointing out sideways. Or, put another way, a fern is just a branch with two rows of smaller ferns stuck alongside the branch. And, of course, eventually we get to some actual leaves.

Here’s a pair of data definitions that capture that intuition.

A fern is either
- (make-leaf number)
- (make-branch number number list-of-fern list-of-fern)

A list-of-fern is either
- empty
- (cons fern list-of-fern)

(define-struct branch (length thickness left right))
(define-struct leaf (area))
List the names of the functions provided by the following definitions:

```
(define-struct branch (length thickness left right))
(define-struct leaf (area))
```

Solution

- `branch?`
- `branch-length`
- `branch-thickness`
- `branch-left`
- `branch-right`
- `leaf?`
- `make-leaf`
- `leaf-area`
Write three example ferns and two example list-of-ferns.

Solution

(make-leaf 10)
(make-branch 10 10 empty)
(make-branch 10 10 (cons (make-leaf 10) empty))

empty
(cons (make-leaf 10) empty)
Write the *fern* template and the *list-of-ferns* template.

**Solution**

```scheme
;; fern-template : fern → ???
(define (fern-template a-fern)
  (cond
    [(leaf? a-fern)
     ... (leaf-area a-fern) ...]
    [(branch? a-fern)
     ... (branch-length a-fern) ...
     ... (branch-thickness a-fern) ...
     ... (list-of-fern-template (branch-left a-fern)) ...
     ... (list-of-fern-template (branch-right a-fern)) ...])

;; list-of-fern-template : list-of-fern → ???
(define (list-of-fern-template a-lof)
  (cond
    [(empty? a-lof) ...]
    [(cons? a-lof)
     ... (fern-template (first a-lof)) ...
     ... (list-of-fern-template (rest a-lof)) ...])
```

4
Imagine the function \textit{weight} that computes the weight of a fern in pounds. Leaves weigh 0.5 pounds per unit area. For a branch, if the length is \(l\) and the thickness is \(t\), the weight (in pounds) is

\[ \pi t^2 l \]

Compute the weights of your example ferns (leaving the answer as a formula in terms of \(\pi\) is fine — that’s what the little man is for, after all).

\textbf{Solution}

\[
\begin{align*}
5 \\
1000\pi \\
1000\pi + 5
\end{align*}
\]
Write the function weight.

Solution

;;; weight : fern → number
(define (weight a-fern)
 (cond
  [(leaf? a-fern)
   (* 0.5 (leaf-area a-fern))]
  [(branch? a-fern)
   (+ (* pi
        (branch-thickness a-fern)
        (branch-thickness a-fern)
        (branch-length a-fern))
     (weight-lof (branch-left a-fern))
     (weight-lof (branch-right a-fern)))]))

;;; weight-lof : list-of-fern → number
(define (weight-lof a-lof)
 (cond
  [(empty? a-lof) 0]
  [(cons? a-lof)
   (+ (weight (first a-lof))
       (weight-lof (rest a-lof)))]))
Here is a data definition for lists of numbers that always have at least one number in them:

A non-empty-lon is either:
- (cons number empty)
- (cons number non-empty-lon)

Write the function \( \text{max} : \text{non-empty-lon} \rightarrow \text{number} \). It computes the maximum number in a list of numbers, \textit{without} assuming that the numbers are all positive.

\textbf{Solution}

\begin{verbatim}
(define (max nelon)
  (cond
    [(empty? (rest nelon)) (first nelon)]
    [else (max-num (first nelon) (max (rest nelon)))]))

;; max-num : number number \rightarrow number
(define (max-num a b)
  (cond
    [(< a b) b]
    [else a]))
\end{verbatim}