We have given you a lot to study. Here are a few more practice questions. They are not exhaustive. These are just some questions we think may be helpful. They certainly do not span the entire space of questions we may ask you.
1. **Trees and Structural Induction.** Some binary trees are heaps. One of the properties of a heap is that it is “heap ordered”: the key stored in each node is smaller than the key stored in any of its children. This heap property can be tested for on a standard binary tree. (It will usually be false for a binary search tree, though.) Write a boolean predicate `heap-ordered?`. Assume that `leaf?` is true when a node has no children (so the empty node is also considered a leaf). Assume that `smaller-child` returns the smaller of a tree node’s children (it is an error if `smaller-child` is called on a leaf.) Assume that `before?` tests for “less-than” when passed two keys that are stored in the tree.

```
(define heap-ordered?
  (lambda ((tr <btree>))
    ...
    A tree which is heap-ordered, will be ordered using the key values. They are obtained as follows:

  Contracts:
  Type `<btree>` is exactly as defined in the notes for Lecture 9 and recitation Section 6. There are no changes from those handouts, but we reproduce them here for your convenience:

  (defclass <btree> ()
    (datum :initarg :datum :accessor datum)
    (left :initarg :left :accessor left)
    (right :initarg :right :accessor right))

  (define leaf?
    (lambda (o) ; o is type `<obj>`
      (not (instance-of? o <btree>))))

  (define-class <datum> ()
    (key :initarg :key :accessor key)
    (contents :initarg :contents :accessor contents))

  (left tr) returns the left subtree of a tree node tr that is not a leaf.
  (right tr) returns the right subtree of a tree node tr that is not a leaf.
  (datum tr) returns the datum stored at tree node tr. “Datum” is the singular of “data” (pl).
  (key (datum tr)) returns key value of the datum stored at tree node tr.

  Write the function “heap-ordered?”
  How do you know it is correct?
```
2. **Set Theory.**

We will represent finite sets as lists. Hence the set \{1, 2, 3, 5\} would be created by
\(\text{list 1 2 4 5}\) and have value \(1\ 2\ 3\ 5\).

We will represent ordered pairs as lists of two elements. Hence, the ordered pair \(5, 2\)
would be created by \(\text{list 5 2}\) and have value \(5\ 2\).

Do not build ordered pairs as \(\text{cons 5 2}\), which would have value \(5 . 2 \neq (5\ 2)\).

We will represent infinite sets as streams.

(a) Suppose \(A\) and \(B\) are sets. The **Cartesian Product** of \(A\) and \(B\) is

\[ A \times B = \{ (a, b) | a \in A, b \in B \} \]

Write a Scheme function \texttt{cart} that computes the Cartesian product of two finite sets \(A\) and \(B\):

\begin{verbatim}
(define cart
  (lambda (A B)
    ...
  ))
\end{verbatim}

(b) Same question, but now assume \(A\) and \(B\) are infinite streams.

(c) Given a set \(A\), the **power set** of \(A\), denoted \(\mathcal{P}(A)\) is the set of all subsets of \(A\).

i. If \(A\) has 2 elements, show that \(\mathcal{P}(A)\) has four elements.

ii. How many elements does \(\mathcal{P}(A)\) have if \(A\) has one element?

iii. Three elements?

iv. No elements?

v. If the set \(A\) has \(n\) elements, how large is the power set of \(A\)?

vi. Prove your answer by induction.

vii. Why is \(\mathcal{P}(A)\) called the power set of \(A\)?