Write code using Scam syntax and semantics unless otherwise directed. Do not use assignment or loop constructs such as while or for unless otherwise directed.

1. Consider this version of cons, which is only valid for small, non-negative, integer arguments:

\[
\text{(define (cons a b)} \text{ (* (^ 7 a) (^ 11 b))})
\]

The \(^\text{\textasciicircum}\) function implements exponentiation and returns an integer if both its arguments are integers. Define a compatible version of \text{car}.

2. Using the style of the text (Exercise 2.6), give the Church numeral representation of \text{four}.

Give a base value and incremener function such that the realization of \text{four} would yield the Scam number 16. In the text, the parameter \(f\) is the incremener and the parameter \(x\) is the base value.

3. Draw a minimal cons structure that represents a singly-linked list with a tail pointer. The linked list contains the numbers 1, 2, and 3 in that order. Have the variable \text{items} point to this linked list.

Give a minimal expression, using cars and cdrs, that retrieves the first value, via the variable \text{items}.

Give a minimal expression, using cars and cdrs, that retrieves the last value, via the variable \text{items}. Make use of the tail pointer.

4. Define a function, named \text{circular?}, that determines whether or not a list wraps back upon itself. That is, does the last cdr point to a cons cell somewhere on the backbone of the list?

\text{MISSING IMAGE}

Hint: walk the list with two pointers, one which takes a single step and one which takes a double step. If the faster-moving pointer reaches \text{nil}, then the list does not have a circular tail. If the faster moving pointer ever equals the slower one, then it must have wrapped around on a cycle to do so. For simplicity, assume the incoming list has an odd number of cons cells.

5. Recall the \text{interleave} function that shuffles two lists.

\[
\text{(define (interleave s t)} \text{ (if (null? s)} \text{ t \text{ (cons (car s) (interleave t (cdr s))})})
\]

Define a version of \text{interleave}, named \text{merge}, that takes two ordered lists and a two-argument predicate function. This predicate function returns true if its first argument \text{precedes} the second. The resulting list should be ordered. Note: the lists are not necessarily made up of numbers. Hence the need for the predicate function.

6. Collect all the pairs \((i, j)\) such that \(0 \leq i < j\) and \(0 \leq j < n\). Pick from the components \text{enumerate}, \text{map}, \text{keep}, \text{remove}, \text{accumulate}, and \text{expand}. The \text{expand} function takes a list, a single item, and a location, and creates a list of lists composed each of the list items and the single item. For example, \text{(expand '(1 4 2) 0 'back)} evaluates to \(((1 0) (4 0) (2 0))\), while \text{(expand '(1 4 2) 0 'front)} evaluates to \(((0 1) (0 4) (0 2))\). The second argument to enumerate is exclusive. Start with \(n\).

7. Define a version of a generic \text{eval} function that evaluates a given parse tree under a given environment using a data-directed approach. Assume evaluation functions for specific parse trees have been stored in a table by parse-tree type. Assume the existence of the following functions: \text{type}, which returns the type of a given parse tree and \text{get}, which retrieves the evaluation function associated with the given type from the table. Do no error checking. Hint: \text{eval} is not a very big function.
8. Define the recognizing function for the following grammar rule:

```
art : INK blue PENCIL
    | green
    | green CHALK
    | green purple OIL
    | indigo indigo violet
    | WATERCOLOR
```

Pseudocode is OK.

9. Define a pending function for the rule shown in the previous question. Pseudocode is OK.