CS115: Module 7 Extra Practice Problems

1. Write a function \(\text{included?} \): \(\text{lst1} \ \text{lst2}\) that consumes two lists \(\text{lst1}\) and \(\text{lst2}\) and produces true if \(\text{lst1}\) is in \(\text{lst2}\) and false otherwise.
   Examples:
   \[
   \begin{align*}
   (\text{included?} \ (1 \ 3 \ 2) \ (1 \ 2 \ 3)) &= \text{true} \\
   (\text{included?} \ \text{empty} \ (1 \ \text{true})) &= \text{true} \\
   (\text{included?} \ (3.5 \ \text{make-posn} \ 3 \ 5) \ (\text{list} \ 10 \ \text{false} \ \text{make-posn} \ 3 \ 5)) &= \text{true}
   \end{align*}
   \]

2. Write a function \(\text{in-occ? val n lst}\) that consumes two integers \(\text{val}\) and \(\text{n}\) and a list of integers \(\text{lst}\) and produces true if \(\text{val}\) appears in \(\text{lst}\) exactly \(\text{n}\) times and false otherwise.
   Examples:
   \[
   \begin{align*}
   (\text{in-occ?} \ 5 \ 3 \ (1 \ 2 \ 5 \ 0 \ 5 \ -10 \ 0 \ 5 \ -1)) &= \text{true} \\
   (\text{in-occ?} \ 10 \ 2 \ (10 \ 10 \ 10)) &= \text{false}
   \end{align*}
   \]

3. Write a function \(\text{list>?} \): \(\text{lst1} \ \text{lst2}\) that consumes two lists \(\text{lst1}\) and \(\text{lst2}\) of the same length and produces a boolean based on the following rule:
   \[
   \begin{align*}
   \text{lst1} &= (\text{list} v1 \ v2 \ \ldots \ vn) \\
   \text{lst2} &= (\text{list} w1 \ w2 \ \ldots \ wn) \\
   v1 &> w1 \ \text{and} \ v2 > w2 \ \text{and} \ \ldots \ \text{and} \ vn > wn \Rightarrow \text{true} \\
   \text{and} \ \text{false} \ \text{otherwise}.
   \end{align*}
   \]
   Example:
   \[
   (\text{list>?} \ (\text{list} \ 10 \ 5 \ -4 \ 0) \ (\text{list} \ 3 \ -10 \ -20 \ 5)) = \text{true}
   \]
   because \(10 > 3, 5 > -10, -4 > -20, \) and \(0 > -5\)

4. Write a function \(\text{cons-list-posn} \): \(\text{lst1} \ \text{lst2}\) that consumes two lists \((\text{listof Num}) \ \text{lst1}\) and \(\text{lst2}\) of the same length and produces a \((\text{listof Posn})\) where the x values of the \text{Posn} will be taken from \(\text{lst1}\) and the y values of the \text{Posn} will be taken from \(\text{lst2}\).
   Example:
   \[
   (\text{cons-list-posn} \ (\text{list} \ 3 \ 2) \ (\text{list} \ -1 \ 0)) = \\
   (\text{list} \ \text{make-posn} \ 3 \ -1) \ (\text{make-posn} \ 2 \ 0))
   \]

5. Write a function \(\text{list-equiv} \): \(\text{lst1} \ \text{lst2}\) that consumes two lists of numbers and produces true if both \(\text{lst1}\) and \(\text{lst2}\) have the same values. Note: Order does not matter and assume no repetition of values in each list.
   Example:
   \[
   (\text{list-equiv} \ (\text{list} \ 1 \ 3.5 \ -10) \ (\text{list} \ 3.5 \ 1 \ -10)) = \text{true}
   \]
The following structure definition will be used in question 6, 7, and 8.

```scheme
(define-struct student (name avg))
(define student1 (make-student "Nisha" 70))
(define student2 (make-student "Bettina" 90))
(define student3 (make-student "Mbabi" 80))
(define student4 (make-student "Judah" 12))
```

6. Write a function `(merge-students los1 los2)` that consumes two sorted lists of students, sorted by name, and produces a new list by merging the contents of `los1` and `los2`.
   Example:
   ```scheme
   (merge-students (list student2 student1) (list student3))
   => (list student2 student3 student1)
   ```

7. Write a function `(sort-students-by-name los)` that consumes a list of students `los` and produces the same list of students sorted by name.
   Example:
   ```scheme
   (sort-students-by-name (list student1 student2 student3 student4))
   => (list student2 student4 student3 student1)
   ```

8. Write a function `(sort-students-by-avg los)` that consumes a list of students `los` and produces the same list of students sorted by `avg` (from smallest to highest).
   Example:
   ```scheme
   (sort-students-by-avg (list student3 student2 student1 student4))
   => (list student4 student1 student3 student2)
   ```

9. Write a function `(shopping consumer retailer)` that consumes two `(listof Str)` and produces a new list containing items from the consumer’s list that appear on the retailer’s list.
   Example:
   ```scheme
   (shopping (list "cereal" "apples" "bananas" "pop")
             (list "apples" "bananas" "cereal" "milk" "potatoes"))
   => (list "cereal" "apples" "bananas")
   ```

10. Write a function `(unique list1 list2)` that consumes two `(listof Nat)` where `list1` must contain sorted non-decreasing numbers between 0 and 4 (inclusive) and `list2` must contain sorted non-decreasing numbers between 5 and 9 (inclusive). `list1` can contain duplicates but `list2` must contain distinct values. `unique` will produce a list containing the numbers from `list1` followed by those from `list2`, but with all the duplicates removed from `list1`.
    Example:
    ```scheme
    (unique (list 0 0 1 2 2 3 3 4 4 4 4) (list 6 7 8 9))
    => (list 0 1 2 3 4 6 7 8 9)
    ```
11. Write a function `(my-lcm n1 n2)` that consumes two `Nat` and produces the lowest common multiple of `n1` and `n2.
Example:
`(my-lcm 81 3465) => 31185`