CS 301: Formal Languages and Functional Programming, Fall, 2016
Programming Assignment 2, due: November 14, 2016

1. Implement a set equality function called set-equal?. The function expects two lists that it interprets as sets. As such, each list in a well-formed input contains no duplicate members but may contain embedded lists. Your function need not check that the input has the correct form, but it should correctly respond #t or #f on well-formed inputs.

2. Implement a Scheme function, RemoveFirst, that takes as arguments a natural number n and a list L and removes the first n elements of L. That is, the function returns a new list containing elements $(n+1)$ through the end of the input list L. If n is greater than the length of L, the function returns the empty list. If n is negative, the function returns #f. For example, `(RemoveFirst 2 '(fee fi fo fum))` returns `'(fo fum)`.

3. Implement a Scheme function, RemoveLast, that takes as arguments a natural number n and a list L. This function is similar to the preceding exercise but removes the *last* n elements of L. Again, if n is greater than the length of L, the function returns the empty list, and if n is negative, the function returns #f. For example, `(RemoveLast 2 '(fee fi fo fum))` returns `'(fee fi)`.

4. Implement a Scheme function, SliceList, that takes as arguments a natural number m, a natural number n, and a list L. The function returns the selected elements in positions m through n as a new list. Assume that the list's first element occupies position 1. Return #f for inapplicable values of m and/or n, such as $m > n$, negative values, or n greater than the length of L. For example,

   `(SliceList 2 3 '(2 3 4 5))` returns `'(3 4)`
   `(SliceList 2 5 '(2 3 4 5 6))` returns `'(3 4 5 6)`
   `(SliceList 2 6 '(2 3 4 5 6))` returns #f.

5. Implement a Scheme function, pair, that takes two lists L and M, and returns a list in which each element is a pair of corresponding elements. For example, `(pair '(fee fi) '(fo fum))` returns `'((fee fo) (fi fum))`. The two lists are required to have the same length. If not, return #f. If both lists are empty, return an empty list.

6. Implement a Scheme function, pair2, that takes two lists L and M, and returns a list in which each element is a pair of corresponding elements. However, the two lists are not required to have the same length. If the length of one is shorter than the other, then the length of the resulting list is the same as the length of the shorter list. If one of the lists is empty list, return an empty list. For example,

   `(pair2 '(fee fi) '(fo fum))` returns `'((fee fo) (fi fum))`
   `(pair2 '(fee fi) '(fo fum fo))` returns `'((fee fo) (fi fum))`
   `(pair2 '(fee fi fi) '(fo fum))` returns `'((fee fo) (fi fum))`.

7. Implement a Scheme function called div that takes two natural number arguments, a dividend and a divisor, and returns a list giving their quotient and remainder. Define this function through recursive use of the subtraction operator '-'. For example, `(div 13 4)` returns `'(3 1)`. The function should return #f if the divisor is zero.