Assignment 2, Fall 2018
Due Friday Nov 9th, 2018
(execute on Racket Scheme download, test before submitting)
Note the changes to the 2017 assignment.

1. **make6** - takes two integers, and returns a 6-digit integer constructed of the leftmost 3 digits of the first input, and the rightmost 3 digits of the second input. For example, (make6 561432 254) would return 561254. Negative signs on either input number should be ignored - that is, (make6 561432 -254) would also return 561254. If the first number has less than three, and/or the second number has less than three digits, your function should return -2. Note: you may want to define some auxiliary functions.

2. **concatL** – takes two lists of lists of the same length and returns a list of that length containing strings which are the concatenation of the strings at the same position in the two list. For example, (**concatL** '((a b) (c d) (d e)) '((f f f) (d e s) (v v v v)) will return ("abfff" "cddes" "devvvv"). Note: you may want to define some auxiliary functions. You may also use the built-in function "string-append" which takes two strings as arguments and returns a string which is the concatenation of the two string arguments.

3. **buildList** – Takes an integer N and a Scheme expression E1 (i.e. an atome or a list) and an integer M and a Scheme Expression E2 and
returns a new list of length M+N where each element is the Expression E1 N times and E2 M times. For example:
(buildList 5 '() 3 'B) will return the list (( ) ( ) ( ) ( ) ( ) B B B)
(buildList 3 'A 2 C) will return (A A A C C)
(buildList 2 '(a b c) 1 Q) will return ((a b c) (a b c) Q)

4. **DFA-Acceptor** - Consider the following DFA which may start with zero or more 1s which must be followed by a 01 which may be followed by any string. The DFA has the following transitions:

Q0 is the start state. P is a sink state for transitions that lead to a fail state (string is not accepted if it transitions to this state P). F = {Q2} is the final state.
Write a DFA that returns **#t** (**True** in Scheme) if it accepts a string and returns **#f** (**False** in Scheme) if it does not accept the string.
Q0 on 1 transitions to Q0
Q0 on 0 transitions to Q1
Q1 on 1 transitions to Q2
Q1 on 0 transitions to P
Q2 on 0 and 1 transitions to Q1.
The DFA stops with a Boolean return value of **#f** on reaching P and returns **#t** on reaching Q2 at the end of the string. Assume the string is represented as a list of 0s and

1s. Write functions corresponding to Q0, Q1, Q2 and P that returns functions. For example the function Q0 when called represents the state it transitions to: (Q0 0) returns the function Q1. (Q0 1) returns the function Q0.

The DFA Acceptor is called as follows:

(DFA-Acceptor <list of alphabets in string to be accepted> <start-state> (<list of final states>) <sink-state>)

(DFA-Acceptor '(1 1 0 1 0) Q0 (Q2) P) returns #t
And
(DFA-Acceptor '(1 0 0) Q0 (Q2) P) returns #f

5. **selectN -** takes as input an integer N. It then builds and returns a "select" function based on N. The "select" function that is produced (by your function selectN) would have the property that it takes as input a list, and returns the same list with the last N elements removed. For example, if selectN was called as follows:
(selectN 3)
a function would be produced that takes as input a list and returns the list with the last 3 elements removed from that list.
For example, if the original call had been made as follows:
(define First (selectN 3))
then the produced function C would behave as follows:
(First '(4 8 2 9 -1 13)) *** would return (4 8 2)
(First '(-2 3 -4 8 9 1 7)) *** would return (-2 3 -4 8)
Your task is just to write selectN.
Of course, selectN should work for ANY input integer, not just 3.