## For example, "Mr. Justin Case" or "Miss Alice Inwonderland".
##
## Write a function new_title which consumes a name
## and produces a new name, which is the same as the first, except
## that if the title was "Miss" or "Mrs.", the new name has
## the title "Ms.".
##
##
## # Problem 4:
## # Write a Python function check_password which consumes a string and a
## # natural number. The function then repeatedly asks the user to input the:
## # password, until it matches the consumed string or until the user has made
## # N incorrect guesses, where N is the consumed number. The function prints
## # Welcome
## # if the password is guessed correctly and it produces True.
## # If the function is not guessed correctly and the user runs out of guesses
## # then
## # Access Denied
## # is printed, and False produced.
##
## # Problem 5
## # Write a Python function draw_triangle that consumes a natural number, n
## # and prints a triangle over n lines, as shown below. The function produces
## # None
## # draw_triangle(3) prints
## # X
## # XX
## # XXX
## #
## # draw_triangle(6) prints
## # X
## # XX
## # XXX
## # XXXX
## # XXXXX
## # XXXXXX
Consider a new type: Table. A Table is a (listof (listof Int)), which is nonempty, and in which each list corresponds to a row of a Table. It is assumed that each row is nonempty and each row has the same number of entries as every other row.

For example, the following are tables:

\[
\begin{align*}
t_0 &= \text{[[1]]} \\
t_1 &= \text{[[1,2,3]]} \\
t_2 &= \text{[[1],[2],[3]]} \\
t_3 &= \text{[[1,2],[3,4],[5,6],[7,8]]} \\
t_4 &= \text{[[1,2,3,4],\ [5,6,7,8],\ [9,10,11,12],\ [13,14,15,16],\ [17,18,19,20]]}
\end{align*}
\]

Write a function \texttt{sum\_columns} that consumes a \texttt{Table} \texttt{t}, and produces a list containing the columns sums for \texttt{t}.

For example,

\[
\begin{align*}
\text{sum\_columns}(t_0) &= \text{[1]} \\
\text{sum\_columns}(t_1) &= \text{[1,2,3]} \\
\text{sum\_columns}(t_3) &= \text{[16, 20]}
\end{align*}
\]
Table.

\[
\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{array}
\]

\[
\frac{12}{12} \quad \frac{15}{15} \quad \frac{18}{18}
\]

\[
\text{sum-columns(T)}
\]

\[
\begin{array}{ccc}
\text{Table} & \text{sum-of-all} \\
\end{array}
\]

\[
\text{without list(table01)}
\]

\[
\text{with list(table01)}
\]

\[
\begin{array}{c}
12 \\
\end{array}
\]
# Problem 3:

# Use accumulative recursion to write the function `spread` that consumes a list of numbers, and produces the difference between the largest and smallest values in the list.  
# For example, `spread([3, 1, 9, 17, -4, 2])` => 21,  
# `spread([2])` => 0, `spread([])` => 0.

#
# Problem 4:
# Use accumulative recursion to write a function `majority` that consumes a list of booleans and determines if there are more True than False values in the list.  
# For example, `majority([True, False, False])` => `False`,  
# `majority([False, True, False, True])` => `False`,  
# `majority([False, True, True, True, True])` => `True`
Spread:

\[ 3 | 0 | 1 | 9 | 17 | -4 \]

\[ \text{min} = 3 \times \frac{1}{4} = -4 \]
\[ \text{max} = 8 \times 17 \]
\[ \text{acc} = 0 \times 28 + 16 \circ 21 \]
T F T
num-true 0
num-false 0 x 2

Base case: If empty list:
return num_true > num_false
$n=1$

$\mathcal{L}_1(L)$

$\mathcal{L}_2(L)$

$\mathcal{L}_3(L)$

$\mathcal{L}_4(L)$

$n=2$

$n=3$

$n=4$
Write a function `selective_add_one` which consumes a list of integers \( L \) and produces a new list \( M \) consisting of all the integers of \( L \) greater than or equal to 7 except each value is also incremented by 1.