Page 370 / 7 Recursive algorithm computing nx, $n \in Z^+$, $x \in Z$, using additions only

procedure *nx_rec* (*n:* positive integer, *x:* integer)
if $n = 0$ **then return** *0*
else return *nx_rec* $(n - 1, x) + x$
{output is $n \cdot x$}

or

procedure *nx* (*n:* positive integer, *x:* integer)
return *nx_helper* $(n, x, 0)$
{output is $n \cdot x$}

procedure *nx_helper* (*n:* positive integer, *x:* integer, *sum*: integer)
if $n = 0$ **then return** *sum*
else return *nx_helper* $(n - 1, x, sum + x)$
{output is $n \cdot x$}

Page 370/10 recursive algorithm for finding the maximum in a finite set of integers

procedure *min_rec* $(L = a_1, \ldots, a_n$ *:* integers)
if $n = 1$ **return** a_1
else
    if $a_{n-1} <= a_n$ **then** $L_1 = a_1, \ldots, a_{n-1}$
    else $L_1 = a_1, \ldots, a_{n-2}, a_n$
    return *min_rec*(L_1)

or

procedure *min* $(L = a_1, \ldots, a_n$ *:* integers)
if $n = 1$ **return** a_1
else return *min_helper* (L, a_n)
{output is the minimum value in the list L}

procedure *min_helper* $(L = a_1, \ldots, a_n$ *:* integers, *min*: integer)
if $n = 0$ **return** *min*
else
    $L_1 := a_1, \ldots, a_{n-1}$
    if $a_n < min$ **then return** *min_helper*(L_1, a_n)
    else return *min_helper*(L_1, min)
{output is the minimum value in the list L}