Matthew McCaughan
Date: 10/5/2022
I pledge my honor that I have abided by the Stevens Honor System.

p. 67, #4 a, b, c, d, e

A:
This algorithm sums all perfect squares from the square of 1 to the square of the integer input

B:
This algorithm initializes a sum, *S*, to zero in which numbers will be added to it. There is a loop where variable i iterates from 1 to *n*, the input value. In this loop, the value of i in the iteration is multiplied to itself and added to *S*, squares of *i* are added to *S* until the loop breaks, and it returns S

C:
The primary operation of the algorithm is executed *n* times, where n is the nonnegative integer input

D:
This algorithm runs at an efficiency of **θ(n)**.

E:
I would argue that this algorithm is very efficient at its job. Repeated addition to a variable cannot be done in one step unless all the values are calculated beforehand, and even then, if they were stored in some array, you would have to iterate through the array to find the value you need. Maybe recursion would be a little more efficient, where Mystery(n) performs only i*i, adds it to *S*, and then calls Mystery(n-1) with base case of n being 1.

p. 76, #1 a, b, c, d, e

A:
x(n) = x(n − 1) + 5 for n > 1, x(1) = 0

Step 1:
x(n-1) = x(n-2) +5
x(n) = **x(n-1)** + 5
x(n) = x(n-2) + 10

Step 2:
x(n-2) = x(n-3) + 10
x(n) = x(n-2) + 10
x(n) = x(n-3) + 15

Step 3:
x(n) = x(n-i) + 5*i

Step 4:
Initial condition: x(1) = 0, so n - i = 1 -> i = n -1

Step 5:
x(n) = x(n-i) + 5*i
x(n) = x(n-n-1) + 5*(n-1)
x(n) = 0 + 5n -5

x(n) = 5n - 5

B:
x(n) = 3x(n − 1) for n > 1, x(1) = 4
Step 1:
x(n-1) = 3x(n-2)
x(n) = 3x(n-1)
 = 3(3x(n-2)
 = 9x(n-2)

Step 2:
x(n-2) = 3x(n-3)
x(n) = 9x(n-2)
 = 9(3x(n-3))
 = 27x(n-3)

Step 3:
x(n) = 2^k * x(n-k)

Step 4:
Initial condition: x(1) =4, so n = k + 1, k = n - 1

Step 5:
x(n) = 2^k * x(n-k)
x(n) = 2^(n-1) * x(n-n-1)
x(n) = 2^n-1 * 4

$x(n) = 2^{(n-1)} * 4$

C:
x(n) = x(n − 1) + n for n > 0, x(0) = 0

Step 1:
x(n-1) = x(n-2) + n-1
x(n) = x(n-2) + n-1 + n

Step 2:
x(n-2) = x(n-3) + n-2
x(n) = x(n-2)+ n-1 + n
x(n) = x(n-3) + (n-2) + n-2) + n

Step 3:
x(n) = x(n-i) + (n-i+1) +(n-i+2)+...+n

Step 4:
Initial condition: x(0) = 0, so, n-i = 0 -> i =n

Step 5:
x(n) = x(n-i) + (n-i+1) +(n-i+2)+...+n
x(n) = 0 + 1 + 2 + 3 + 4 + … + n = n(n+1) / 2

x(n) = n(n+1) / 2

D:
$x(n) = x(n/2) + n$ for $n > 1$, $x(1) = 1$ (solve for $n = 2k$)

Step 0:
$x(2^k) = x(2^{k-1}) + 2^k$

Step 1:
$x(2^{k-1}) = x(2^{k-2}) + 2^{k-1}$
$x(2^k) = x(2^{k-1}) + 2^k$
$\quad = x(2^{k-2}) + 2^{k-1} + 2^k$

Step 2:
$x(2^{k-2}) = x(2^{k-3}) + 2^{k-2}$
$x(2^k) = x(2^{k-2}) + 2^{k-1} + 2^k$
$\quad = x(2^{k-3}) + 2^{k-2} + 2^{k-1} + 2^k$

Step 3:
$x(2^k) = x(2^{k-i}) + 2^{k-i+1} + 2^{k-i+2} + \dots + 2^k$

Step 4:
$2^{k-i} = 1$
$2^{k-i} = 2^0$
$k-i = 0$
$i = k$

Step 5:
$x(2^k) = x(2^{k-i}) + 2^{k-i+1} + 2^{k-i+2} + \dots + 2^k$
$x(2^k) = x(2^{k-k}) + 2^{k-k+1} + 2^{k-k+2} + \dots + 2^k$
$\quad = x(2^0) + 2^1 + 2^2 + \dots + 2^k$
$\quad = 1 + 2^1 + 2^2 + \dots + 2^k$
$n = 2^k$
$\mathbf{x(n) = 1 + 2^1 + 2^2 + \dots + n}$
$$\mathbf{x(n) = (2^{n+1} - 1) + n}$$

E:
$x(n) = x(n/3) + 1$ for $n > 1$, $x(1) = 1$ (solve for $n = 3^k$)

Step 0:
$x(3^k) = x(3^{k-1}) + 1$

Step 1:
$x(3^{k-1}) = (3^{k-2}) + 1$
$x(3^k) = \mathbf{x(3^{k-1}) + 1}$
$x(3^k) = x(3^{k-2}) + 1 + 1$
$x(3^k) = x(3^{k-2}) + 2$

Step 2:
$x(3^{k-2}) = x(3^{k-3}) +1$
$x(3^k) = x(3^{k-2}) + 2$
$= x(3^{k-3}) + 1 + 1 +1$
$= x(3^{k-3}) + 3$

Step 3:
$x(3^k) = x(3^{k-i}) + i$

Step 4:
$3^{k-i} = 1$
$3^{k-i} = 3^0$
$k - i = 0$
$i = k$

Step 5:
$x(3^k) = x(3^{k-i}) + i$
$= x\ (3^{k-k}) + k$
$= x(1) + k$
$= 1 + k$

$n = 2^k$
$k = \log(n)$

$x(n) = 1 + \log(n)$

$x(n) = 1 + \log(n)$

p. 76-77, #3 a, b

A:
S(n) = S(n-1) + n*n*n,   S(1) = 1

Step 1:
S(n-1) = S(n-2) + (n-1)*(n-1)*(n-1)
S(n) = S(n-1) + n*n*n
S(n) = S(n-2) + (n-1)*(n-1)*(n-1) + n*n*n

Step 2:
S(n-2) = S(n-3) + (n-2)*(n-2)*(n-2)
S(n) = S(n-2) + (n-1)*(n-1)*(n-1) + n*n*n
S(n) = S(n-3) + (n-2)*(n-2)*(n-2) + (n-1)*(n-1)*(n-1) + n*n*n

Step 3:
$S(n) = S(n-i) + (n-i + 1)^3 + (n-i+2)^3+...+(n)^3$

Step 4:
Initial condition: S(1) = 1, so n - i = 1 -> i = n-1

$S(n) = S(n-i) + (n-i + 1)^3 + (n-i+2)^3+...+(n)^3$
$S(n) = S(n-n-1) + (n - n - 1 + 1)^3 + (n-n-1 + 2)^3 + … + (n)^3$
$= 0 + 1 + 2^3 … (n)^3$
This algorithm executes its basic operation n times (from 1 up to n).

B: This algorithm is on par with the straightforward nonrursive algorithm for sumputing this sum, the nonrecursive variant uses a loop that iterates from 1 to n, so the same amount of iterations take place between both variants of this algorithm. Both operate at an efficiency of θ(n).