Natalie Zoladkiewicz
CS 385 Homework 2
I pledge my honor that I have abided by the Stevens Honor System.

4a. The algorithm computes the sum of the squares of n numbers.

4b. Multiplication

4c. The algorithm does 1 multiplication per loop. Therefore, if the function is of size n, the algorithm does multiplication n times.

4d. $\Theta(n)$

4e. One way to improve this algorithm is by starting with n and following an algorithm that does not care how many inputs a user provides. If we do $\frac{n(n+1)(2n+1)}{6}$, with the value of the last n, we will get the same result and time complexity will be $\Theta(1)$.

7a.
$$
\begin{aligned}
x(n) &= x(n-1)+5 \quad \text{for } n>1,\ x(1)=0 \\
&= [x(n-2)+5]+5 = x(n-2)+5.2 \\
&= [x(n-3)+5]+5.2 = x(n-3)+5.3 \\
&= [x(n-i)+5]+5i \\
&= x(1)+5.(n-1) \\
&= 0+5(n-1) \\
&= 5(n-1)
\end{aligned}
$$

7b.
$$
\begin{aligned}
x(n) &= 3x(n-1), \text{ for } n>1,\ x(1)=4 \\
&= 3[3x(n-2)] = 3^2x(n-2) \\
&= 3(3[3x(n-3)]) = 3^3x(n-3) \\
&= 3^ix(n-i) \\
&= 3^{n-1}x(n-i) \\
&= 3^{n-1}x(1) \\
&= 4.3^{n-1}
\end{aligned}
$$

7c.
$$
\begin{aligned}
x(n) &= x(n-1)+n, \text{ for } n>0,\ x(0)=0 \\
&= [x(n-2)+(n-1)]+n = x(n-2)+(n-1)+n \\
&= [x(n-3)+(n-2)+(n-1)]+n = x(n-3)+(n-2)+(n-1)+n \\
&= [x(n-i)+(n-i+1)+(n-i+2)+\ldots+n \\
&= x(0)+1+2+3+\ldots+n = \frac{n(n+1)}{2}
\end{aligned}
$$

7d. $x(n) = x(\frac{n}{2}) + n$, for $n>1$, $x(1)=1$

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

$= [x(2^{k-2}) + 2^{k-1}] + 2^{k} = x(2^{k-2}) + 2^{k-1} + 2^{k}$

$= x(2^{k-i}) + 2^{k-i+1} + \ldots + 2^{k}$

$= 1 + 2^{1} + 2^{2} + \ldots + 2^{k}$

$= 2^{k+1} - 1$

$= 2.2^{k} - 1$

$= 2n-1$

7e. $x(n) = x(\frac{n}{3}) + 1$, for $n>1$, $x(1)=1$

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

$= [x(3^{k-2}) + 1] + 2 = x(3^{k-3}) + 3$

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

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

$= x(1) + k$

$= 1 + \log_3 n$

3a. I am using $x(n)$ to denote the number of multiplications made.

$x(n) = x(n-1) + 2$

Solve where $x(1) = 0$:

$[x(n-2)+2] + 2 = x(n-2) + 2 + 2$

$[x(n-3)+2] + 2 + 2 = x(n-3) + 2 + 2 + 2$

$x(n-i) + 2i$

$x(1) + 2(n-1)$

$= 2(n-1)$

3b. The nonrecursive version would look something like this:

input: any integer n that is $n>0$
output: sum of first n cubes

```
S ← 1
for i ← 2 until n do:
    S ← S + i·i i
return S
```

The only difference is that the non-recursive version doesn't have as much space usage as the recursive