1. **Find the runtime of the algorithm mathematically (I should see summations).**

```
function x = f(n)
        x = 1;
        for i = 1:n
                for j = 1:n
                 x = x + 1;
```

Answer: Total operations = n*n=n^2

So, the runtime of the algorithm in terms of the number of operations (x = x + 1; statements executed) is O(n2)O(n^2)O(n^2).

∑i=1n∑j=1n1=n*n=n^2

The runtime of the algorithm is O(n^2)

3. **Find polynomials that are upper and lower bounds on your curve from #2. From this specify a big-O, a big-Omega, and what big-theta is.**

Answer: Upper Bound (Big-O): O(n^2)
Lower Bound (Big-Omega): Omega(n^2)
Tight Bound (Big-Theta): Θ(n^#2)

4. Will this increase how long it takes the algorithm to run (e.x. syou are timing the function like in #2)?

If I modified the function to be:

```
x = f(n)
  x = 1;
  y = 1;
  for i = 1:n
     for j = 1:n
```

```
x = x + 1;
y = i + j;
```

Answer: The additional line of code will slightly increase the algorithm's runtime, but the impact is minimal and negligible since it introduces only a constant time operation within the inner loop.