Vinay Kankanala
1002210909

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

1. Find the runtime of the algorithm mathematically

<u>Inner loop</u>

It executes n times for each phase in outer loop.

∴ n executions for inner loop

<u>Outer loop</u>

It executes n times from $i=1$ to $i=n$

∴ n executions for outer loop

Total Execution $= \sum_{i=1}^{n} \sum_{j=1}^{n} 1$

$= \sum_{i=1}^{n} n$

$= n \times n$

$= n^2$

∴ The total runtime of algorithm is $O(n^2)$

3) Find Polynomials that are upper and lower bounds on your curve: From this specify a big-O, big-omega, and what big-theta is.

Big-O-natation (upper bound)

$O(n^2)$-function does not grow faster than Quadrate

Big-omega

~~O~~ $(\Omega(n^2))$ -function does not grow slower than Quadrate

Big-Theta

$O(n^2)$ -function grows asymptotically as n^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 = y + j;
```

~~Vinay Kankanala~~

~~1002210909~~

4) will this increate how long it takes the algorithm to run?

$y = i+j$, actual time taken by the modified function will be slightly longer due to it.

$\therefore$ $O(n^2)$ is the time complexity

5) will it effect your result from first func?

It doesn't effect much in the time complexity.