(3.1) Given a time Function, $T(n) = 2n^2 + 4n + 3$, find Constants C and N that prove that the Big O of the growth function $T(n)$ is n^2.

$T(n) = 2n^2 + 4n + 3$ $\quad C * g(n)$

$2n^2 + 4n + 3 <= 2n^2 + 4n^2 + 3n^2$

$2n^2 + 4n + 3 <= 10n^2$

$C = 10 \quad n_0 = 1$ ✓

Guessing $O(n^2)$ for time Complexity.

Find The Big O of $T(n) = 2n^3 + 5n^2 + n + 2$. Justify your answer by finding Constants C and N.

Guessing $O(n^3)$ for time Complexity.

(3.2) $T(n) = 2n^3 + 5n^2 + n + 2$

$2n^3 + 5n^2 + n + 2 <= 2n^3 + 5n^3 + n^3 + 2n^3$

$2n^3 + 5n^2 + n + 2 <= 10n^3$

In the form $C * g(n)$

$C = 10, \quad n_0 = 1$

Big O: $O(n^3)$

3.3 Find the Big O of the following code:

```
for(int i=0, i<n/2; i++)
  {
    for(int j=0; j<n/2; j++)
    {
    System.out.println("Hello!");
    }
}
```

The Big O is: $O(n^2)$

The inner loop can be expressed as $2n+2$, and for every iteration of the outer loop (equal to n times) there's a full n iterations from the inner loop:

$(2n+2) * N = 2n^2 + 2n$

This "nested loop" is an n^2 time complexity contribution.