2. Show directly that $f(n) = n^2 + 3n^3 \in \Theta(n^3)$. That is, use the definitions of O and Ω to show that f(n) is in both $O(n^3)$ and $\Omega(n^3)$.

Part A).

We need to show that there exist positive constants c and n0 such that:

f(n)≤ c x n^3 for all n≥n0

Using f(n) = n^2 + 3n^3, we compare it to n^3:

n^2 + 3n^3 ≤ 4n^3 for n≥1

Choosing c=4 and n0=1, we satisfy the Big-O condition.

Thus, f(n) ∈ O(n^3)

Part B).

We need to show that there exist positive constants c' and n0'' such that:

f(n) ≥ c′ x n^3 for all n ≥ n0′

Since f(n) = n^2 + 3n^3, the dominant term is 3n^3, so:

n^2+3n^3 ≥ 3n^3 for all n≥1

Choosing c′=3 and n0′=1, we satisfy the Big-Omega condition.

Thus, f(n)∈Ω(n^3)

3. Show that $5n^5 + 4n^4 + 6n^3 + 2n^2 + n + 7 \in \Theta(n^5)$.

To show that $5n^5 + 4n^4 + 6n^3 + 2n^2 + n + 7 \in \Theta(n^5)$ we need to prove that a positive c and n0, such that f(n) >= c * n^5 for all n >= n0. Since $f(n) = 5n^5 + 4n^4 + 6n^3 + 2n^2 + n + 7$, the most prominent term is $5n^5$. Therefore $5n^5 + 4n^4 + 6n^3 + 2n^2 + n + 7 >= 5n^5$ for all n >= 1. Choosing c = 5 and n0 = 1, so we satisfy the big-O condition. Therefore $f(n) \in O(n^5)$.

4. The function $f(x) = 3n^2+10n \log n+1000 n+4 \log n+9999$ belongs in which of the following complexity categories:

(a) Θ(lg n) (b) $\Theta(n^2 \log n)$ (c) Θ(n) (d) Θ(n lg n)
(e) $\Theta(n^2)$ (f) None of these

Hint: See properties 6 and 7 and Example 1.23 on pages 39 and 40 of the textbook for a very similar example.

- The dominant term is $3n^2$ and it grows faster than all of the other terms.(n^2 > nlog(n) > n > log(n) > constant; via asymptotic complexity) Thus, the the complexity of this function is $\Theta(n^2)$.

5. Consider the following algorithm:

```
for( i = 1 ; i <= 1 . 5 n ; i++)
    cout << i;
for( i = n ; i >= 1 ; i - - )
    cout << i;
```

(a) What is the output when n = 2, n = 4, and n = 6?

(b) What is the time complexity T(n)? You may assume that the input n is divisible by 2.

- (a)
  - i) n = 2
    - Upper for-loop: 1 / 2 / 3
    - Lower for-loop: 2 / 1
    - Output: 12321
  - ii) n = 4
    - Upper for-loop: 1 / 2 / 3 / 4 / 5 / 6
    - Lower for-loop: 4 / 3 / 2 / 1
    - Output: 1234564321
  - iii) n = 6
    - Upper for-loop: 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9
    - Lower for-loop: 6 / 5 / 4 / 3 / 2 / 1
    - Output: 123456789654321
- (b)
  - Upper for-loop: The variable i starts at 1 to 1.5n, so it is O(n)
  - Lower for-loop: The variable i starts at 1 to n, so it is O(n)
  - O(n) + O(n) = O(n)
  - Thus, the time complexity is $\Theta(n)$