Q1 :

1- $2^{n+1} = 2^n * 2 \longrightarrow O(2^n)$

2- $2^{2n} = (2^n)(2^n) = (2^n)^2 \quad (2^n)^2 \neq O(2^n)$

3- $(\frac{1}{4})^n \neq O(n)$

Q2 : Prove that $F(n) = \theta(g(n))$

1- $3n^3$ is the biggest term

So $F(n) = 3n^3 \longrightarrow \theta(n^3)$

$\therefore F(n) = \theta(n^3)$

2- $F(n) = 2^{n+1} = 2^n * 2 \longrightarrow \theta(2^n)$

$\therefore F(n) = \theta(2^n)$

3- $\log(n)$ is the biggest term

$F(n) = \log(n) \longrightarrow \theta(\log(n))$

$g(n) = \ln(n) = \log_e n \longrightarrow \theta(\log(n))$

$\therefore F(n) = \theta(g(n))$

Q3 : Prove that $F(n) = o(g(n))$ or $F(n) = \omega(g(n))$

1- $F(n) = n^3$, $g(n) = n^2$

$n^3 > n^2 \longrightarrow F(n) \geq c\,g(n)$

$\therefore F(n) = \omega(g(n))$

2- $F(n) = \log(n)$, $g(n) = \log^2(n)$

$\log(n) < \log^2(n) \longrightarrow F_n \leq c\,g_n$

$\therefore F(n) = O(g(n))$

Q4 : $C_1 g(n) \leq T(n) \leq C_2 g(n)$, $n \geq n_0$

IF the running time of the algorithm is $\theta(g(n))$

∴ the best case : $T(n) \geq C_1 g(n)$, $n \geq n_0 \longrightarrow T(n) = \Omega(g(n))$

∴ the worst case : $T(n) \leq C_2 g(n)$, $n \geq n_0 \longrightarrow T(n) = O(g(n))$

Q5: $O(g(n)) \longrightarrow$ $T(n) \leq C g(n)$ For $n \geq n_0$

and this indicates that $T(n)$ is bounded ~~below~~ above by $g(n)$ $n > n_0$

$w(g(n)) \longrightarrow$ $T(n) \geq C g(n)$ For $n \geq n_0$

and this indicates that $T(n)$ is bounded ~~above~~ below by $g(n)$ $n > n_0$

So $O(g(n)) \cap w(g(n))$ is empty set