Q1:

1. Is $2^{n+1} = O(2^{n})$?
2. Is $2^{2n} = O(2^{n})$?
3. Is $(0.25)^{n} = O(n)$?

1-yes--------> $O(2^{n}\times 2)=O(2^{n})$ neglect constant

2-no --------> $O(2^{2n})=O((2^{n})^{2}) \neq O(2^{n})$

3- no --------> $O(0.25^{n})=O((1/4)^{n}) \neq O(n)$

Q2:

For every given $f(n)$ and $g(n)$ prove that $f(n) = \boldsymbol{\theta}(g(n))$

1. $g(n) = n^{3}$, $f(n) = 3n^{3} + n^{2} + n$
2. $g(n) = 2^{n}$, $f(n) = 2^{n+1}$
3. $g(n) = ln(n)$, $f(n) = log(n) + log(log(n))$

1- ($\mathcal{F}(n)=3n^{3} + n^{2} + n) \leq \text{const}\times n^{3}$ So :$g(n)=n^{3}$

2- ($\mathcal{F}(n)= 2^{n+1}) \leq \text{const}\times 2^{n}$ So :$g(n)= 2^{n}$

3- ($\mathcal{F}(n)= \log(n)+ \log(\log(n))$) $\leq \log(n) \times \ln(10) = \ln(n)$---->

$\mathcal{F}(n)=\Theta(\log(n))$ or $\Theta(\ln(n))$

So :$g(n)= \ln(n)$

Q3:

For every given $f(n)$ and $g(n)$ prove that $f(n) = \boldsymbol{O}(g(n))$ or $f(n) = \boldsymbol{\Omega}(g(n))$

1. $f(n) = n^3$, $g(n) = n^2$
2. $f(n) = log(n)$, $g(n) = log^2(n)$

1-Let's check $\mathscr{F}(n)=O(g(n))$

($\mathscr{F}(n)=n^3$) $\leq const\times n^2$ -----> $n \leq const$ ----> $n \to \infty$ so condition doesn't met

For large n, this inequality does not hold for any constant c. Therefore $\mathscr{F}(n) \neq O(g(n))$

Let's check $\mathscr{F}(n)=\Omega(g(n))$

($\mathscr{F}(n)=n^3$) $\geq const\times n^2$

This inequality holds for $n \geq c$ with $c>0$. So :$\mathscr{F}(n)=\Omega(g(n))$

2-Let's check $\mathscr{F}(n)=O(g(n))$

($\mathscr{F}(n)= log(n)$) $\leq$ ($const\times log^2(n) = const\times 2\times log(n)$)

$1 \leq c\times log(n)$

For $n \geq 2$ choosing $c=1$ satisfies the inequality. So : $\mathscr{F}(n)=O(g(n))$

Q4:

Prove that the running time of an algorithm is $\boldsymbol{\theta}(g(n))$ if and only if its worst-case running time is $\boldsymbol{O}(g(n))$ and its best-case running time is $\boldsymbol{\Omega}(g(n))$

By definition, the running time of an algorithm is $\Theta(g(n))$ if and only if there exist constants

$c_1, c_2 > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$

Average case: $C_1\times g(n) \leq \mathscr{F}(n) \leq C_2\times g(n)$

Upper bond is worst case $O(g(n))$ [$C_1\times g(n) \leq \mathscr{F}(n)$]

&

lower band is best case $\Omega(g(n))$ [$\mathscr{F}(n) \leq C_2\times g(n)$]

Q5:

Prove that $\boldsymbol{O}(g(n)) \cap \boldsymbol{\omega}(g(n))$ is the empty set.

- $f(n) = \omega(g(n))$ mean [$C_2>0$ and $n_0 \geq 0$ such that for all $n \geq n_0$ $f(n) > C_2 \times g(n)$]
- $f(n) = O(g(n))$ mean [$C_1>0$ and $n_0 \geq 0$ such that for all $n \geq n_1$ $f(n) \leq C_1 \times g(n)$]

  $C=\max(C_1, C_2)$ $n_2=\max(n_0, n_1)$.

  $O(g(n)) \cap \omega(g(n)) = \emptyset$ → $f(n)$ cannot be both less than and greater than $C \times g(n)$ for the same n.