Q2)

a)

Big-O notation means the worst case scenario which also means the maximum time that alghoritm can take. So "The running time of algorithm A is at least $O(n^2)$" means "minimum of the maximum time that taken by alghoritm A is n^2. Big-O notation used for upper-bound but "at least" is a lower-bound term. That's why it is meaningles to say.

b)

Say f(n) = $n^2 + n$ and $g(n) = n$,
max(f (n), g(n)) = f(n) so $f(n) = \theta(n^2)$
$\theta(f(n) + g(n)) = \theta(n^2 + n + n) = \theta(n^2)$. So it is true.

Say f(n) = n and $g(n) = n^2 + n$,
max(f (n), g(n)) = g(n) so $g(n) = \theta(n^2)$
$\theta(f(n) + g(n)) = \theta(n^2 + n + n) = \theta(n^2)$. So it is true.

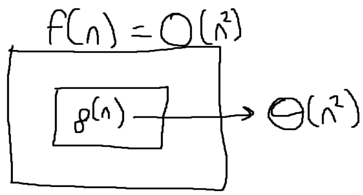
Say f(n) = n and $g(n) = n$
max(f (n), g(n)) = f(n) or g(n).Let's say f(n). So $f(n) = \theta(n)$
$\theta(f(n) + g(n)) = \theta(n + n) = \theta(n)$. So it is still true.

And there is no other situation left. So its all above proves that max(f (n), g(n)) = θ(f(n) + g(n))

b) i) $\lim_{n\to\infty} \frac{2^{n+1}}{2^n} = 2$
So $\lim_{n\to\infty} \frac{2^{n+1}}{2^n}$ equals to a constant (c).
So $f(n) = \theta(g(n))$, (g(n) is 2^n and f(n) is 2^{n+1}).
So it is correct.

ii) $\lim_{n\to\infty} \frac{2^{2n}}{2^n} = 2^n = \infty$
So $f(n) = \Omega(g(n))$, (f(n) is 2^{2n} and g(n) is 2^n).
So it is false.

iii)



It is wrong, it must be $O(n^4)$ because we don't know about f(n). It could be quadratic or constant or something else.

Q3)
$3^n >$ is the greatest because it's base is 3 , the greatest number among others. Then,
If n is greater than or equal to 2 , $2^{n+1} > n2^n$ because it's base is lower than 3.Else, $n2^n > 2^{n+1}$ Then,
$5^{\log_2 n} > 2^n$ because once in two $5^{\log_2 n}$ acts like 5^n which is still greater than 2^n. Then,
$n^{1.01} > \sqrt{n}$ because $n^{1.01}$ is very close to being linear.Then,
If $n > \log n$, $nlogn > \log n^3$. Then,
log n is the smallest growing term. Because the logorithm grows the smallest.