4. (a) f(x) + g(x) is o(f(x)g(x))? - **False** This is because there exists examples such as $g(x) = \frac{1}{x}, f(x) = x$ where:

$$\lim_{x\to\infty} \frac{f(x)+g(x)}{f(x)g(x)}$$

will not tend towards 0 but instead trend upwards:

$$\frac{x+\frac{1}{x}}{x*\frac{1}{x}}$$
$$= x + \frac{1}{x}$$
$$\lim_{x\to\infty} x + \frac{1}{x} = \infty$$

(b) $2^x * x^2$ is $o(2.1^x)$? - **True**

$$\lim_{x\to\infty} \frac{2^x * x^2 + (2.1)^x}{2^x * x^2 * (2.1)^x}$$
$$= \lim_{x\to\infty} \frac{2^x * x^2}{2^x * x^2 * (2.1)^x} + \frac{(2.1)^x}{2^x * x^2 * (2.1)^x}$$
$$= \lim_{x\to\infty} \frac{1}{(2.1)^x} + \frac{1}{2^x * x^2}$$
$$= \lim_{x\to\infty} \frac{1}{(2.1)^x} + \lim_{x\to\infty} \frac{1}{2^x * x^2}$$
$$= \frac{1}{\infty} + \frac{1}{\infty}$$

Which tends towards 0 meaning the claim is true.

(c) $x^2log(x)$ is $O(x^2)$? - **False** This is false as $x^2log(x)$ grows faster than x^2. There exists no pair of constant witnesses (C, k) that will satisfy the following inequality:

$$|C * x^2| \geq |x^2log(x)|, x \geq k$$

This is simply because the function $log(x)$ will obviously, eventually, always be larger than any constant C.

(d) $x^2log(x)$ is $O(x^3)$? - **True** Firstly:

$$x^3 = x^2 * x$$
$$|x| \geq |log(x)|, x \geq 1$$

This shows the function x grows faster than the function $log(x)$ and x^3 is a product of this faster growing x function and x^2 meaning the statement is true:

$$|x^3| \geq |x^2log(x)|, x \geq 1$$

(e) $7x^5$ is $O(12x^4 + 5x^3 + 8)$? - **False**
Firstly, we know:

$$|x^5| \geq |x^4| \geq |x^3| \geq |x^2| \geq |x| \geq |1|, x \geq 1$$

Effectively, the claim is false as the x with the greatest degree will always be the dominant term as x increases. This means $7x^5$ grows faster than $12x^4 + 5x^3 + 8$ so it cannot be $O(12x^4 + 5x^3 + 8)$.

5. (a) $T(n) = 64T(\frac{n}{8}) - n^2 * log_2(n)$

$$|f(n)| = n^2 * log(n)$$
$$a = 64$$
$$b = 8$$
$$log_b(a) = log_8(64) = 2$$

Case 2 of the master theorem applies as f(n) $= \theta(n^{log_b(a)} * log^k(n)) : k = 1$
$\therefore$ T(n) $= \theta(n^2 * log^2(n))$.

(b) $T(n) = 4T(\frac{n}{2}) + \frac{n}{log(n)}$

$$f(n) = \frac{n}{log(n)} = O(n)$$
$$a = 4$$
$$b = 2$$
$$log_b(a) = log_2(4) = 2$$

$\frac{n}{log(n)} = O(n)$ as n will obviously grow faster than $\frac{n}{log(n)}$ for C $=$ 1, k $=$ 0. Therefore, case 1 of the master theorem applies as f(n) $= O(n^{log_b(a) - \epsilon}) : \epsilon = 1$.
$\therefore$ T(n) $= \theta(n^2)$.

(c) $T(n) = 2^n T(\frac{n}{2}) + n^2$ Here, the master theorem cannot be applied as the recurrence is not in the form $T(n) = aT(\frac{n}{b}) + f(n)$ (a and b **must** be constants).

(d) $T(n) = 3T(\frac{n}{4}) + n * log(n)$

$$f(n) = nlog(n)$$
$$a = 3$$
$$b = 4$$
$$log_b(a) = log_4(3)$$

$f(n) = \Omega(n)$ for $k > 2$ and $C = 1$ so we can apply case 3 of the master theorem as $fn(n) = \Omega(n^{log_4(3)-\epsilon})$: $\epsilon = (1 - log_4(3))$ to get $f(n) = \Omega(n)$. Furthermore we most show the following holds for some c:

$$3\frac{n}{4}log(\frac{n}{4}) \leq c * nlogn$$
$$c = \frac{3}{4}$$

$\therefore$ using case 3 of the master theorem we deduce that $T(n) = \theta(nlogn)$.

(e) $T(n) = 3T(\frac{n}{3}) + \sqrt{n}$

$$f(n) = \sqrt{n} = n^{\frac{1}{2}}$$
$$a = 3$$
$$b = 3$$
$$log_b(a) = log_3(3) = 1$$

Therefore, case 1 of the master theorem applies as $f(n) = O(n^{log_b(a)-\epsilon})$: $\epsilon = \frac{1}{2}$ to get $f(n) = O(n^{\frac{1}{2}})$.
$\therefore$ T(n) = $\theta(n)$

6. (b) The worst-case input structure of this algorithm will be when the list is sorted. This is the case since we pick the pivots to be the k many rightmost elements of the 'unsorted' list. In this structure of input, this will be the k greatest/smallest elements of list. Essentially, his means we will create just 2 groups - the pivot and 'less/more than the pivot (depending on if the list is sorted or reverse sorted respectively)'.

Next when sorting the 'not the pivot' side it will continue picking the worst pivot (effectively the smallest or greatest number left in the list) everytime. Since we are going over the list n times then traversing it again n times every time we get a worst-case time complexity of O(n^2).