1. (a) f = Θ(g). To verify this, we must prove that f = O(g) as well as f = Ω(g). First, when c = 2, f(n) <= 2g(n) for all n >=100. Then when c = 1, f(n) >= g(n) for all n >=0;
   (b) $n^{1/2} = O(n^{2/3})$. This is simply because $n^{1/2}$ is always smaller than $n^{2/3}$ no matter how small the c constant is, given that x is large enough.
   (c) f = Θ(g). Since both f and g = O(n logn), f = O(g) and g = O(f) when c is changing.
   (d) f = Θ(g). To verify this, we must prove that f = O(g) as well as f = Ω(g). As for O, when c = 1, obviously, f(n) grows slower than g(n). Therefore, f = O(g). Similarly, when f = Ω(g), let's say c = 0.001. In this case f(n) = nlogn, c g(n) = 0.001nlog10n. Same as what is done in the previous approach, f(n) surely grows faster than g(n).
   (e) f = Θ(g). Since both of f and g = O(n logn), f = O(g) and g = O(f) when c is changing.
   (f) f = Θ(g). Since both of f and g = O(n logn), f = O(g) and g = O(f) when c is changing.
   (g) f(n) = Ω(g). f can be simplified as $n*n^{0.01}$ while g = $n*(\log n)^2$. In this way, we can find that $n^{0.01}$ is superior to $\log^2 n$. Therefore f(n) = Ω(g).
   (h) If we multiply both f and g by logn/n, we have f = n and g = $(\log n)^3$. And there is no doubt that a power function is superior to the cubic of a logarithmic function. Therefore, f = Ω(g)
   (i) Same as what is illustrated in the (h), f = Ω(g)
   (j) f = Ω(g). Since f(n) can be simplified as f(n) = $n^{\log\log n}$, f becomes a power function which means f always wins.
   (k) f is a power function which means it always grows faster than g(n). Therefore, f = Ω(g)
   (l) g can be simplified as g(n) = $n^{\log_2 5}$ > f(n) = $n^{1/2}$. Therefore, f = O(g).
   (m) f = O(g). Since 2^n is dominated by 3^n with the definition of the exponential function.
   (n) f = Θ(n). Because f and g both = O(2^n).
   (o) f = Ω(g). Because a factorial function grows much faster than an exponential function.
   (p) f = O(g). Since f(n) can be simplified as f(n) = $n^{\log\log n}$, g(n) can be simplified as $n^{\log_2 n}$, obviously f(n) grows faster than g(n).
   (q) f = O(g). Since f = 1 + 2^k + 3^k + ... + n^k. g = n^k + n^k+ ……(n times), f grows slower than g.
2. (a) g(n) = $(c^{n+1} -1)/(c-1)$. When n approaches the positive infinite, lim g(n) =1/(1-c). And 1/(1-c)>1. But if 1 * a constant c 1/(1-c) can be smaller than 1*c. Therefore g(n) = Θ(1).
   (b) when c = 1. g(n) = n +1 = Θ(n)
   (c) when c > 1 lim g(n) = $c^{n+1}/(c-1) = \Theta(c^n)$
4. (a) Let's say a matrix A = a b c d, the other B = e f g h. In this case A×B = ae + bg af + bh ce + dg cd + dh. And we have done 4 additions and 8 multiplications. Thus proved.
   To calculate X^n, it takes n matrix multiplications.
   (b) $X^n = X^{(n/2)} * X^{(n/2)}$ (n is even)
   $X^n = X * X^{(n/2)} * X^{(n/2)}$ (n is odd)
   We can see that the whole process would take logn(n is even) or 1+logn(n is odd) multiplications. Therefore, O(logn) matrix multiplications suffice for computing X^n.