* 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) n1/2 = O(n2/3). This is simply because n ½ is always smaller than n2/3 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\*n0.01 while g = n\*(logn)2. In this way, we can find that n0.01 is superior to log2n. Therefore f(n) = Ω(g).

(h) If we multiply both f and g by logn/n, we have f = n and g = (logn)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 loglogn, 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) = nlog25 > f(n) = n1/2. Therefore, f = O(g).

(m) f = O(g). Since 2n is dominated by 3n with the definition of the exponential function.

(n) f = 𝛩(n). Because f and g both = O(2n).

(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 loglogn, g(n) can be simplified as nlog2n, obviously f(n) grows faster than g(n).

(q) f = O(g). Since f = 1 + 2k + 3k + … + nk. g = nk + nk+ ……(n times), f grows slower than g.

2. (a) g(n) = (cn+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) = 𝛩(cn)

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 Xn, it takes n matrix multiplications.

(b) Xn = X (n/2) \* X (n/2) (n is even)

Xn = 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 Xn.