Week I and Week II problem Set:

Part I:

① $f(n) = O(g(n))$ implies $g(n) = O(f(n))$

② $f(n) + g(n) = \Theta(\min(f(n), g(n)))$

③ $f(n) = O(g(n))$ implies $\lg(f(n)) = O(\lg(g(n)))$, where $\lg(g(n)) \geq 1$ and $f(n) \geq 1$ for all sufficiently large n.

④ $f(n) = O(g(n))$ implies $2^{f(n)} = O(2^{g(n)})$.

⑤ $f(n) = O((f(n))^2)$.

⑥ $f(n) = O(g(n))$ implies $g(n) = \Omega(f(n))$.

⑦ $f(n) = \Theta(f(n/2))$.

⑧ $f(n) + o(f(n)) = \Theta(f(n))$.

Part II:

Ordering by asymptotic growth rates:

→ Rank the following functions by the order of growth: that is, find an arrangement $g_1, g_2, \dots, g_{30}$ of the function satisfying $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, $\dots$, $g_{29} = \Omega(g_{30})$. Partition your list into equivalence classes such that $f(n)$ and $g(n)$ are in the same classes if and only if $f(n) = \Theta(g(n))$.

a) $\lg(\lg^* n)$ $\quad 2^{\lg^* n}$ $\quad (\sqrt{2})^{\lg n}$ $\quad n^2$ $\quad n!$ $\quad (\lg n)!$

b) $(3/2)^n$ $\quad n^3$ $\quad \lg^2 n$ $\quad \lg(n!)$ $\quad 2^{2^n}$ $\quad n^{1/\lg n}$

c) $\ln\ln n$ $\quad \lg^* n$ $\quad n \cdot 2^n$ $\quad n^{\lg\lg n}$ $\quad \ln n$ $\quad 1$

d) $2^{\lg n}$ $\quad (\lg n)^{\lg n}$ $\quad e^n$ $\quad 4^{\lg n}$ $\quad (n+1)!$ $\quad \sqrt{\lg n}$

e) $\lg^*(\lg n)$ $\quad 2^{\sqrt{2\lg n}}$ $\quad n$ $\quad 2^n$ $\quad n\lg n$ $\quad 2^{2^{n+1}}$

Part III:

↳ Solve the Recurrence Relation for the following:

a) $T(n) = 1T\left(\frac{n}{2}\right) + K$

b) $T(n) = 4T\left(\frac{n}{2}\right) + O(n)$

c) $T(n) = 3T\left(\frac{n}{2}\right) + O(n)$

d) $T(n) = 8T\left(\frac{n}{2}\right) + c \cdot n^2$

e) $T(n) = 7T\left(\frac{n}{2}\right) + c \cdot n^2$

f) $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$

g) $T(n) = T\left(\frac{3n}{10}\right) + T\left(\frac{7n}{10}\right) + O(n)$

h) $T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{2n}{3}\right) + n^2$.

Part IV

1) What is the Complexity of the program given below:

```
Void function (int n) {
    int i, j, k, count = 0;
    for (i = n/2 ; i <= n ; i++)
        for ( j = 1 ; j <= n ; j = 2*j)
            for ( k = 1 ; k <= n ; k = k*2 )
                Count ++;
}.
```

2) Write a recursive function for the runningtime $T(n)$ of the functi-on given below. Prove using the iterative method that $T(n) = \Theta(n^3$

```
function (int n) {
    if (n == 1) return;
    for (int i = 1 ; i <= n; i++)
        for (int j = 1 ; j <= n ; j++)
            Print ("*");
    function (n-3);
}.
```

3) Determine Θ bounds for the recurrence relation:

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + n.$$

4) What is the complexity of $\sum_{i=1}^{n} \log i$?

5) Write a recursion formula for the runningtime $T(n)$ of the function whose code is below:

```
function (int n) {
    if (n <= 1) return;
    for (int 1 = 1 ; i < n ; i++)
        Printf ("*");
    function (0.8n);
}.
```

6) xyz (A, l, h) (Analyze the running Time).

```
{
  if (l < h)
  {
      t = ⌊ (3l + 2h) / 5 ⌋
    xyz( A, l, t);
    xyz( A, t+1, h);
    xyz( A, l, t, h); =>
  }
}
```

7) Analyze the running time of the following recursive pseudo-code as a function of n.

```
void function (int n) {
        if (n < 2) return;
        else counter = 0;
        for i = 1 to 8 do
                function (n/2);
        for i = 1 to n^3 do
                Counter = Counter + 1;
}.
```

8) find the complexity of following function :

```
Void function (int n){
    if (n<=1) return ;
    if (n>1) {
        Printf ("*");
        function (n/2);
        function (n/2);
    }
}.
```