Tutorial-2

Ans-1:

```
void fun (int n)
{
int j=1, i=0;
while (i<n)
{
  i = i+j
  i++;
}
}
```

$j=1$, $i=0+1$

$j=2$, $i=0+1+2$

$j=3$, $i=0+1+2+3$

Loop ends when $i \geq n$

$0+1+2+3 \ldots n > n$

$\frac{K(K+1)}{2} > n$

$K^2 > n$

$K > \sqrt{n}$

$O(\sqrt{n})$

Ans-2: Recurrence Relation for Fibonaci series

$$T(n) = T(n-1) + T(n-2)$$

$T(0) = T(1) = 1$

- if $T(n-1) \simeq T(n-2)$

$$T(n) = 2T(n-2)$$

(Lower Bound) $= 2\{2T(n-4)\} = 4T(n-4)$

$= 4(2T(n-6))$

$= 8T(n-6)$

$= 8(2T(n-8)$

$= 16T(n-8)$

$\vdots$

$$T(n) = 2^k T(n-2k)$$

$n - 2^k = 0$

$n = 2^k$

$k = \frac{n}{2}$ $\quad T(n) = 2^{n/2}\, T(0)$

$= 2^{n/2}$

$T(n) = \Omega(2^{n/2})$

- if $T(n-2) \simeq T(n-1)$

$T(n) = 2T(n-1)$

$= 2(2T(n-2)) = 4T(n-2)$

$= 4(2T(n-3)) = 8T(n-3)$

$= 2^k\, T(n-k)$

$n - k = 0$

$\boxed{k = n}$

$T(n) = 2^k\, T(0) = 2^n$

$= T(n) = O(2^n)$ (upper bound)

Ans-3: $O(n(\log n))$ =>

```
for (int i=0 ; i<n ; i++)
{
  for (int j =1 ; j<n; j = j*2)
  {
    // Some O(1)
  }
}
```

- $O(n^3)$ =>

```
for (int i=0; i<n ; i++)
{
  for (int j = 0 ; j<n ; j++)
  {
    for (int k=0 ; k<n; k++)
    {
      // Some O(1)
    }
  }
}
```

- $O(\log(\log n))$ ⇒

```
for (int i=1; i<=n; i=i*2)
{
    int j=1; j<=n; j=j*2)
    {
        //some O(1)
    }
}
```

Ans-4: $T(n) = T(n/4) + T(n/2) + cn^2$

Lets assume $T(n/2) >= T(n/4)$

So, $T(n) = 2T(n/2) + cn^2$

applying master's Theorem $T(n) = aT(\frac{n}{b}) + f(n)$

$a = 2, b = 2$ $\quad f(n) = n^2$

$c = \log_b a = \log_2 2 = 1$

$n^c = n$

Compare n^c and $f(n) = n^2$

$f(n) > n^c$ $\quad$ So, $T(n) = \Theta(n^2)$

Ans-5:

```
int fun (int n)
{
    for(int i=1; i<=n; i++)
    {
        for (int j=1; j<n; j+=1)
        {
            // Some O(1)
        }
    }
}
```

$i = 1$ — $j = 1, 2, 3, \ldots, n$ — n times.

$i = 2$ — $j = 1$
$j = 3$
$j = 5$
$j = 7$

— Loop ends when $j > n$

$1 + 3 + 5 + 7 > n$

$k > \frac{n}{2}$

— n times

$i = 3$ — $j = 1$
$j = 4$
$j = 7$

—— $1 + 4 + 7 > n$

$k > \frac{n}{3}$

$i = 4$ —— $k > \frac{n}{4}$

$i = n$ ⋮

So total compar $= O(n^2 + n^2 + n^2 \ldots)$

$= O(n^2)$

Ans - 6: for (int $i = 2$; $i <= n$; $i =$ Pow (i, x))

{

// Some ()

}

Complexity of Pow (i, x) — $O(\log N)$

$= \log(x)$

$i = 2$

$i = 2^x$

$i = 2x^2$

$i = 2x^3$

$i = 2x^4$

⋮

$i = 2x^n$

Loop ends when $i > n$

$$2^{k^n} > n$$

$$\log(2^{k^n}) > \log n$$

$$k^n \log 2 > \log n$$

$$k^n > \log n$$

$$\log(k^n) > \log(\log n)$$

$$n \log k > \log(\log n)$$

$$n > \frac{\log(\log n)}{\log k}$$

$$T(n) = O(\log(\log n))$$

Ans-8: a) $1 < \log n < \sqrt{n} < n < \log(\log n) < n \log n < \log n! < n! < n^2 < \log 2^n < 2^n < 2^{2n} < 4^n$

b) $1 < \sqrt{\log n} < \log n < 2\log n < \log 2N < N < 2N < 4N < \log(\log N) < N \log N < \log N! < N! < N^2 < 2 \times 2^N$

c) $96 < \log_8 N < \log_2 N < n \log_6 N < n \log_2 N < \log n! < N! < 5N < 8N^2 < 7N^3 < 8^{2^n}$