Tutorial-2

Ans-1

```
void func(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 $i \geq n$

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

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

$k^2 > n$

$k > \sqrt{n}$

$O(n)$

Ans-2 Recurrence Relation for fibonacci series

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

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

If $T(n-1) \approx T(n-2)$

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

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

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

$= 8T(n-6)$

$= 16T(n-8)$

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

$n-2k=0$

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

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

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

$$T(n) = 2T(n-1)$$
$$= 2(2T(n-2)) = 4T(n-2)$$
$$= 4(2T(n-3)) = 8T(n-3)$$
$$= 2^k T(n-k)$$

$n-k=0$

$k=n$

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

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

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)
{
  for (int j=1; j<=n; j=j*2)
  {
    ~~for (int ...;~~      // 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 $(T(n) = 2T(\frac{n}{b}) + f(n))$

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

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

$n^c = n$

compare n^c & $f(n) = n^2$

$f(n) > n^c$ 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+=i)
    {
      // somO(1)
    }
  }
}
```

$i=1$ — $j=1$, $j=2$, $j=3$, ⋮, $j=n$ — n times

$i=2$ — $j=1$, $j=3$, ⋮, $j=n$ — loop ends when $j>n$ $1+3+5+\dots k>n$ $k>n/2$

$i=3$ — $j=1$, $j=4$, $j=7$, ⋮, $j=n$ — $1+4+7\dots>n$ $k>n/3$

So total complexity $O(n^2 + n^2 + \dots +)$

$= O(n^2)$

Ans-6

```
for (int i = 2; i <= n; i = pow(i, k))
{
   // some(1)
}
```

complexity pow(i, k) — $O(\log n)$

$= \log(k)$

$i = 2$

$i = 2^k$

$i = 2^{k^2}$

$i = 2^{k^3}$

$\vdots$

$i = 2^{k^m}$

loop ~~wh~~ ends when $i > n$

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

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

$$k^m \log_2 > \log n$$

$$k^m > \log n$$

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

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

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

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

Ans-8 a) a) $100 < \log n < \sqrt{n} < n < \log(\log n) < n\log n < \log n! < n! < n^2 < \log^{2n} < 2^{n}$
$2^{2n} < 4^n$

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

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