ASSIGNMENT:-02

Ans 01>

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

$j=0+1$; $i=0+1$

$j=2$; $j=0+1+2$

$j=3$; $j=0+1+2+3$

$j=n$; $k=n$ HW

$0+1+2+\cdots n > w$

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

$K^2 > w \quad K > \sqrt{n}$

$T(n) = O(\sqrt{n})$

Ans:

Ans 02> Recurrence rel^n for Fibonacci series:-

$T(n) = T(n-1) + T(n-2)$ $\quad T(0) = T(1) = 1.$

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

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

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

$= 4[2T(n-6)] = 8T(n-6)$

$\vdots$

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

$n-2k=0$

$n=2k$

$k = n/2$

$T(n) = 2^{n/2} T(0)$

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

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

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

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

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

$\vdots$

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

$n-1 \le 0$

$k = n$

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

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

$O(n) = 2^n$ Ans:

Ans 03)

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

$\Big\}$ — $O(n \log n)$.

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

$\Big\}$ — $O(n^3)$.

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

$\Big\}$ — $O(\log(\log n))$.

Ans 04)

$T(n) = T(n/4) + T(n/2) + Cn^2$

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

so; $T = 2T(n/2) + Cn^2$

Applying Master's theorm;

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

$c = \log_b a = 1$.

$n^c = n$

Comparing;

$f(n) > n^1$

so, $\boxed{T(n) = \Theta(n^2).}$ — Ans

Ans 05.)

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

i =	1	2	3	4	...
j =	1	1	1	1	...
	2	3	4	5	...
	3	5	7	9	...
	...	...	...	...	...
	n	n/2	n/3	n/4	...

] — n^2

Hence, $T(n) = O(n^2) + O(n^2/2) + O(n^2/3) + \cdots$

$\boxed{T(n) = O(n^2)}$ — Ans.

Ans 06.)

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

pow(i,k) = O(log n)
= log k

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

$\log 2^{k^n} > \log n$ — taking log both sides.

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

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

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

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

Hence;

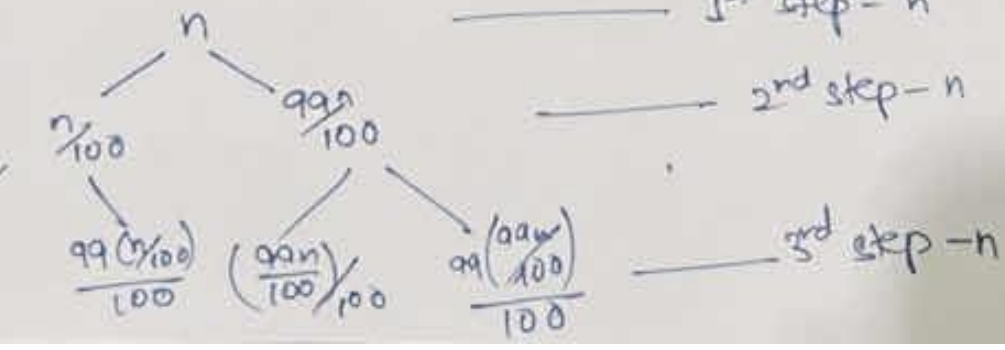
$\boxed{T(n) = \log(\log_2 n)}$

Ans

Ans 07.) Considering the statement;

$T(n) = T(n/100) + T(99n/100) + O(n)$

where $n/100$ & $99n/100$ are for parts & $O(n)$ is partitioning algo.



So; it will remain n at each step.

So; time complexity = $O(n * \log_{100/99} n)$ if we take longer branch

$= \Omega(n * \log_{10} n)$ — TIME COMPLEXITY.

Question 08)

a) Order is :-

$1 < \log n < \sqrt{n} < n < \log(\log n) < n \log n < \log n! < n! < n^2 < \log 2n < 2^n < 4^n$.

(b) Order is.-

$1 < \sqrt{\log n} < \log n < 2 \log n < \log_2 N < N < 2N < 4N < \log(\log N) < N \log N < \log N! < N! < N^2 < 2 \times 2^N$.

(c) Order is..-

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