__/__/____

Tutorial-2.

Ques 1)

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

i	j
0	1
0+1	2
0+1+2	3
0+1+2+3	4
0+1+2+3+4	5
0+1 --- K	K

Assume

$i = K(K+1)/2$

$K(K+1)/2 = n$

$K = \sqrt{n}$

$O(\sqrt{n})$

Ques 2) Recursive fibonacci:

```
int fib(n)
{ if (n<=1)
      return n;
  return fib(n-1) + fib(n-2);
}
```

$T(n) = \underbrace{T(n-1)}_{x} + \underbrace{T(n-2)}_{y}$

$a = 1,$

$x = n$

Let the constants be c

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

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

Here $a = 2$, $b = 1$ and $f(n) = n^0$

$\therefore$ T.C $= O(2^n)$

as we have only one statement to execute we can draw a recursion tree, ~~their~~ height of tree will be the n.

So, space complexity is $O(n)$

Que. 3>

i) T.C $= n \log n$

```
for (i=0; i<n; i++)  ⟶ n
{
    for (j=0; j<n; j*=2)  — log n
        statement;
}
```

Time complexity : $n \log n$

ii> T.C $= n^3$

$T(n) = 8T(n/2) + n$ — $O(n^3)$

```
void fun (int n)
{
    if (n>1)
{
    fun (n/2); fun (n/2); fun (n/2);
    fun (n/2);
    fun (n/2);
    fun (n/2);
```

__/__/____

```
fun(n/2);
fun(n/2);
}
for(i=0; i<n; i++)  ——— n
    stmt;
}
```

$T(n) = \theta(n/2) + n$

$O(n^3)$

iii) $\log(\log n)$

Ques 4) $T(n) = T(n/4) + T(n/2) + cn^2$

$T(n/2) >= T(n/4)$ i.e. $T(n/4)$ is more effective term than $T(n/4)$.

$\therefore T(n) = T(n/2) + T(n/2) + cn^2$

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

master theorem for dividing fun.

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

$n^k = k=2$

$\log_b^a = \log_2^2 = 1 < k$

$O(n^2)$

__/__/____

Ques 5)

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

T.C = $n + n/2 + n/3 + \cdots n/n$

$n\left(\frac{1}{1} + \frac{1}{2} + \frac{1}{3} \cdots + 1/n\right)$

T.C = $n \log n$

Ques 6)

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

$i \to 2, 2^k, (2^k)^k \cdots 2^{k^{\log_k(\log_k n)}}$

Let $i = n$

$2^{k \log_k \log_k n} = n$

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