Tutorial 2

A.1.)

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

	0	1	2	3		n-1
i	1	2	5	7	. . .	n(n-1)
j	1	2	3	4	. . .	n

$i = 1, 3, 5, 7 \ldots n(n-1)$

T(C) of j i.e $(1, 2, 3, 4 \ldots n) = O(n)$

sum of AP i.e $(i) = \frac{n(n+1)}{2}$

$\Rightarrow \frac{n(n+1)}{2} + n(\text{for } j) < n$

$\Rightarrow n^2+n+2n < 2n$

$\Rightarrow n^2 < 2n \Rightarrow n^2 < n$

$\Rightarrow n < \sqrt{n}$

$\therefore T(C) := O(\sqrt{n})$ Ans

A.2.) Fibonacci series program using Recursion:-

```
#include <bits/stdc++.h>
using namespace std;
int fib(int n)
{
  if(n<=1)
```

```
    return n;
    return fib(n-1) + fib(n-2);
}
int main()
{
    int n = 9;
    cout << fib(n);
    getchar();
    return 0;
}
```

$T(n) = T(n)$ i.e linear

Q.3) Write programs which have complexity
(i) $n(\log n)$, n^3, $\log(\log n)$

A.3.) i) $n(\log n)$:-

```
for(i=0; i<n; i++)                    n
{
    for(j=1; j<n; j=j*2)              n x log n
    {
    }
}
```

→ $n(\log n)$

ii) n^3:- Floyd Warshall Algorithm for solving all shortest path flow vertices

i-e for k=1 to n
for i=1 to n
" j=1 to n
$A^k[i,j] = \min(A^{k-1}[i,j], A^{k-1}[i,k] + A^{k-1}[k,j])$
return A

iii) for (i=1 ; i<n; i=i*2) ($T(C) = \log(\log(n))$
{
p++;
}
for (j=1; j<p; j=j*2)
{

}

A.4) T(C) of $T(n) = T(n/4) + T(n/2) + cn^2$

assuming $T(n/2) >= T(n/4)$
On Rewriting
$T(n) <= 2T(n/2) + cn^2$
Now apply master's theorem to RHS
$T(n) <= \theta(n^2) \Rightarrow T(n) = O(n^2)$.

$T(n) >= cn^2 \Rightarrow T(n) >= \theta(n^2) \Rightarrow T(n) = \Omega$
$\therefore T(n) = O(n^2)$ & $T(n) = \Omega(n^2)$
$\Rightarrow T(n) = \theta(n^2)$ Ans

Q.5) What is time complexity of following fun()?

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

A.5) for i=1, inner loop executes n times.
i=2, " " n/2 times
i=3, " " n/3 times
i=n, " " n/n times.

$$T(C) \text{ of } \left(n + \frac{n}{2} + \frac{n}{3} + \ldots + n/n\right)$$

$$\Rightarrow n * \left(1/1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}\right)$$

$$= O(\log n) \text{ Ans}$$

Q.6) What should be time complexity of

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

where k = constant

DATE

A.6) i takes values $2, 2^k, (2^k)^k = 2^{k^2}$

$$\Rightarrow (2^{k^2})^k = 2^{k^3} \ldots 2^{k^{\log_k(\log(n))}}$$

Last term must $<= n$

$$\Rightarrow 2^{k^{\log_k(\log(n))}} = 2^{\log(n)} = n.$$

↓ eqv to last term.

∴ there are $\log_k(\log(n))$

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

Q.7) Write recursive relation . . . . . what do you understand by analysis?

A.7) $$T(n) = T\left(\frac{9n}{10}\right) + T(n/10) + O(n).$$

taking one branch 99% & other 1%.

$$T(n) = T\left(\frac{99n}{100}\right) + T\left(\frac{n}{100}\right) + O(n)$$

1st level = n.

$$2^{nd} \text{ level} = \frac{99}{100} + \frac{n}{100} = n$$

So 3rd level remains same for any kind of position.

Large branch $= O(n \log_{100/99} n)$

for shortest branch

$\hookrightarrow = \Omega(n \log_{10} n)$

either base complexity of $O(n(\log n))$ remains.

A.8.)

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

b.) $K\log(\log n) < \sqrt{\log n} < \log n < \log 2n < n < n\log n =$
$< 2n < 4n = 2(2^n) < n! < n^3$

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