Q1

Ans Asymptotic ~~Anal~~ Notations are mathematical tools to represent the time complexity of algorithms for asymptotic analysis.
Asymptotic Notation describes the running time of an algorithm for a given input.

It is used to analyze the efficiency of the algorithm that is machine-independent.

Mainly Three Notations are:

1) Big-o Notation (O)

2) Omega Notation (Ω)

3) Theta Notation (Θ)

- Big o Notation: Represents the upper bound of the running time of an algorithm.

Therefore, it gives the worst case complexity of an algorithm.

Example - Insertion Sort.

- Omega Notation: Represents the lower bound of the running ~~t~~ time of an algorithm.

Thus, it provides the best case complexity of an algorithm.

- Theta Notations:

It represents the upper and the lower bound of the running time of an algorithm, it is used for analyzing the average-case Time complexity.

Q2 (Ans) Whenever we see a loop whose counter is either being multiplied or divided by any constant value we can be sure to say that its time complexity is $O(\log(n))$ where base of the algorithm is the constant value.

So: $O(\log 2(n))$ is the Time Complexity.

Q3

Ans

$$T(n) = 3T(n-1)$$

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

$$= 3^2 T(n-2)$$

$$= 3^3 T(n-3)$$

Generalised form of equation

$$= 3^n T(n-n)$$

$$= 3^n T(0)$$

$$= 3^n$$

Time Complexity $= O(3^n)$

Page No.
Date
YOUVA

Q4
Ans

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

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

$= 2^2 [T(n-2)] - 2 - 1$

$= 2^2(2T(n-3)-1)-2-1$

$= 2^3 (T(n-3) - 2^2 - 2^1 - 2^0$

- Generalised form of equation

So,:

$2^n T(n-n) - 2^{n-1} - 2^{n-2} - 2^{n-3}$

$\ldots 2^2 - 2^1 - 2^0$

$= 2^n - \cancel{n} (2^n - 1)$

Time Complexity $= O(1)$.

Q5

Ans We can define the terms 's' according to relation $S_i = S_{i-1} + i$.

The value of 'i' increases by one for each iteration

If k is total no. of iterations taken by the program then:

Time Complexity $= O(\sqrt{n})$.

Q6

```
Void function (int n)
{
    int i, Count = 0;
    for(i=1 ; i*i <= n ; i++)
    {
        Count ++;
    }
}
```

$1, 2^2, 3^2, 4^2, 5^2 \dots k^{th}$

k^{th} term $= k * k$

k^{th} term $<= n$

$k * k <= n$

$k^2 = n$

$k = \sqrt{n}$

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

M T W T F S S
Page No.:
Date:
YOUVA

07
Ans

```
Void function (int n)
{
        int i, j, k, Count = 0;
    for (i = n/2 ; i <= n; i++)
    {
            for( j = 1 ; j <= n; j = j+2)
            {
                for (k = 1; k <= n; k = k+2)
                {
                        Count ++;
                }
            }
    }
}
```

Time Complexity of inner most loop

$k = 1$ to n , $k = k + 2$

$1, 2, 4, 8, 16 \ldots$ k^{th} term

k^{th} term $= 2^{k-1}$

$n = \frac{2^k}{2} \Rightarrow 2n = 2^k - 1$

Taking $\log_2$ both Sides

$\log_2 2n = \log_2 2k$

$\log_2 2n = k$

$k = \log_2 2 + \log_2 n$

Page No.: Date: YOUVA

09

$K = 1 + \log_2 n$

Complexity of Middle loop

$j = 1$ to n; $j = j + 2$

$\Rightarrow (1 + \log_2 n)$

Time Complexity of outer most loop:

$i = n/2$ to n ; $i{+}{+}$

$n/2$, $\frac{n}{2} + 1$, $\frac{n}{2} + 2$, $\frac{n}{2} + 3$ ... k^{th}

k^{th} term $= \frac{n}{2} + k$

$\cancel{n}$ $n = \frac{n}{2} + k$

$k = n - \frac{n}{2}$

$= \frac{n}{2}$

$T(n) = O(n\{\log_2 n\}^2)$

Q9

```
Void function(int n)
{
    for(i=1 to n)
    {
        for(j=1; j<=n; j=j+i)
        {
            printf("*");
        }
    }
}
```

Other loop will run n times

for (i=1; j will run n times)

for i=2; j will run n/2 times

for i=3; j will run n/n times

Inner loop will run $= \left(n + \frac{n}{2} + \frac{n}{3} + \cdots \frac{n}{n-1} + \frac{n}{n} \right)$

$$\Rightarrow n \left(1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots \frac{1}{n} \right)$$

$$\Rightarrow n.\log n \Rightarrow O(n \log n)$$

Ans : For fun$(n-3)$

$n, n-3, n-6, n-9$ --- k^{th} term

$n = n-1\cdot 3, n-2\cdot 3, n-3\cdot 3$ -- k^{th} term

k^{th} term $= n-(k-1)-3 \Rightarrow n-3\cdot 3$

$1 = \cancel{n-3k} = n-3k-3$

$\Rightarrow n-3k-k-1=0$

$n-3k-4=0$

$k = \frac{n-4}{3}$

$T(n) = O(n^3)$