Page No.:
Date:

Tutorial - 1

Q1

→ Asymptotic notation is used to describe the running time of an algorithm. Or asymptotic notation tells how much time an algorithm takes to compute wrt the given input.

— Types of asymptotic notations
(a) Big O notation (O)
(b) Big theta notation (Θ)
(c) Big Omega (Ω)

→ Big O → worst case
Big Ω → best case or best running time
Big Θ → used in cases where running time is same in all cases (worst, best).

Q2

→
```
for(i=1 to n)
  { i=i*2; }
```
time complexity = $O(\log_2 n)$
as the value of "i" is incremented by a factor of 2

Q3

→ $$T(n) = \begin{cases} 3T(n-1), & n>0 \\ 1 \end{cases}$$

Q4

— $$T(n) = \begin{cases} 2T(n-1)-1, & n>0 \\ 1 \end{cases}$$

Q5 time complexity of:

```
int i=1;
int s=1;
while(s<=n){
i++;
s=s+i;
printf("#");
}
```

→ value of s → 1, 3, 6, 10 ..... z = n

loop executes z times (suppose)

the final value of s will be k.

$$\frac{z(z+1)}{2} = n$$

$$\frac{z^2+z}{2} = n$$

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

Q6 complexity of:

```
void func(int n)
{ int i, count=0;
  for(int i=1; i*i<=n; i++)
    count++;
}
```

→ O(n)

Page No.:
Date:

Q7 .

→ time complexity → $O(n \log n)$

Q8 time complexity

```
fun(int n)
{ if(n==1) return;
  for(i=1 to n)
    for(j=1 to n)
      printf("*");
  fun(n-3);
}
```

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

Q9

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

Time complexity = $O(n^2)$

Q10 $n^k = O(c^n)$