Page No. ______
Date

Assignment-01

Q1 The asymptotic notation are used to tell the complexity of an algo when the input is very large.

Types of notation -

1) Big-oh (O)
2) Big-omega (Ω)
3) Theta (Θ)
4) Small-oh (o)
5) Small-omega (ω)

Q2 for ($i=1$; $i<n$; $i=i*2$)

- $O(\log n)$

Q3 $T(n) = 3T(n-1) \quad n>0$

$T(1) = 1$

$T(2) = 3T(1) = 3$

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

$T(4) = 3(T(3)) = 3^3 T(n-3)$

$3^0 + 3^1 + 3^2 + 3^3 \text{ - - - - } 3^n T(n-n)$

$= O(3^n)$

Page No.
Date

Q4 $T(n) = 2T(n-1) - 1$

$T(1) = 1$

$\rightarrow T(2(T(n-2)-1)) - 1$

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

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

$\vdots$

$= 2^n T(n-n) - 2^{n-1} - 2^{n-2} \ldots 2^2 - 2^1 - 2^0$

$= \cancel{\dfrac{2^n\left(-\frac{1}{2}^{n+1}+1\right)}{\frac{1}{2}+1}} \Rightarrow \cancel{2^n\left(\dfrac{1+2^{n+1}}{2^{n+1}} \times \dfrac{2}{3}\right)}$

$\Rightarrow \cancel{\dfrac{2^{n+1}}{3} + \dfrac{1}{3}}$

$\cancel{\dfrac{1}{3}(2^n \cdot 2 + 2^n)}$

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

$= T(1)$

$\cancel{2^n = k}$

$\cancel{n = \log 2}$

Q5

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

$\Rightarrow$ $\cancel{T(n) = 1+2+3 \ldots n}$

$= \cancel{\dfrac{n(n+1)}{2}} > k$

n^2

$S_i = S_{i-1} + i$

value of i increase by one on each iteration.

Page No. ____
Date ____

The value of s at the i^{th} iteration is sum of first i^{th} terms.

If k is total no. of iteration.

So,

$$1+2+3+4\ldots+k = \frac{k(k+1)}{2} > n$$

$$k = \sqrt{n}$$

$$\boxed{T(\sqrt{n})}$$

Q6
```
void fun^n (int n)
{ int i, count=0;
  for(i=1; i*i <=n; i++)   // Executes (n^2) times
      count++;
}
```

$$\boxed{O(n^2)}$$

Q7
```
void fun (int n){
  int i,j,k, count=0;
  for(i=n/2 ; i<=n ; i++)           // (n/2)
   for(j=1; j<=n; j*=2)             // (logn)
    for(k=1 ; k<=2 ; k=k*2)         // (logn)
    {
        c++;
}
```

$$O\left(\frac{n}{2} \times \log n \times \log n\right)$$

$$O\left(n \log^2 n\right)$$

Page No.
Date

Q8 fun (int n) {
if (n == 1) return;
for (i = 1 to n)
{ for (j = 1 to n)
{ print (*);
}}
fun (n-3);
}

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

Q9 void fun (int n) {
for (i = 1 to n)
for (j = i ~~2~~ ; j <= n ; j++)
{
Print ("*")
}}

$T(n) = n * (n + (n-1) + (n-2) \ldots 1) = \frac{n(n+1)}{2}$

$\cancel{n+n}$ $\cancel{\Theta(n^2)}$

$T(\cancel{2n})$ $O(n^3)$