Ans1 - Asymptotic notations are used in to find the complexity of an algorithm when input is very large.

$O(o)$: $f(n) = O(g(n))$

if $f(n) \leq c\, g(n)$ i.e $n \geq n_0$

for some constant $c > 0$

$g(n)$ is "Tight upper bound" of $f(n)$

- Big omega (Ω) : $f(n) = \Omega\, g(n)$

  if

  $f(n) \geq c\, g(n)$ i.e $n \geq n_0$

  for some constant $c > 0$

  $g(n)$ is "tight lower bound" of $f(n)$

Big theta (θ) :

$f(n) = \theta(g(n))$

if $c_1\, g(n) \leq f(n) \leq c_2 g(n)$

$\forall\ n \geq \max(n_1, n_2)$

for some constant $c_1 > 0$ and $c_2 > 0$

$g(n)$ is both "tight upper bound and lower bound" of $f(n)$

Ans 2. for (i = 1 to n) { i = i*2 ; }

$1, 2, 4, 8 \ldots n.$

let k^{th} term $= n$

$n = 1(2^{k-1})$

taking log on both side

$\log n = (k-1) \log 2$

$K = \log n + 1$

$O(1 + \log n)$

$O(\log n)$

Ans: 3 $T(n) = 3T(n-1)$ — 1

$n = n-1$ in eqn. 1

$T(n-1) = 3T(n-2)$ — 2

put 2 in 1

$T(n) = 9T(n-2)$ — 3

put
$n = n-2$ in eqn

$T(n-2) = 3T(n-3)$ — 4

now putting this in eqn 3

$T(n) = 27(n-3)$

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

$n - k = 0$

$k = n$

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

$T(n) = 3^n$

$= O(3^n)$

Q4.

$T(n) = 2T(n-1)$ — (1)

$n = n-1$

$T(n-1) = 2T(n-2)$ — (2)

$T(n) = 4T(n-2)$ — (3)

$n = n-2$

$T(n-2) = 2T(n-3)$ — (4)

by eqn. (3) & (4)

$T(n) = 8T(n-3)$

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

$n-k=0$

$k=n$

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

$T(n) = 2^n$

$O(2^n)$

Q5.

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

$i = 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \ldots$

$s = 1 + 3 + 6 + 10 + 15 + \cdots + n$

sum of $s = 1 + 3 + 6 + 10 \cdots + n$ — (1)

$s = 1 + 3 + 6 + 10 + \cdots n-1 + n$ — (2)

from (1) - (11)

$0 = 1 + 2 + 3 + \cdots n - n.$

$T_k = 1+2+3+ \cdots k$

$T_k = \frac{1}{2} k(k+1)$

for k

$1 + 2 + 3 + \cdots + k \leq n$

$\frac{k(k+1)}{2} \leq n$

$(k^2 + k)/2 \leq n$

$O(k^2) \leq n$

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

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

$\underline{O(n^{1/2})}$

Ans 6.

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

$O(1 + \sqrt{n} + \sqrt{n} + \sqrt{n})$

$O(1 + 3\sqrt{n})$

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

7.

```
void func^n (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 ++;
  }
```

i	j	k
n/2	1	1
/	↓	2
n/2	log n	4
		⋮
		n — log n

$O(n/2 \times \log \times \log n)$

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

Q8.

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

n   i   j

$1 + 4 + 7 + \cdots n$

$n = 1 + 3(k-1)$

$= 3k - 2$

$k = \frac{n+2}{3}$

no of term

$\frac{n+2}{6}\left[2 + \frac{(n-1)}{3} \times 3\right]$

$\left[\frac{n+2}{6} \cdot (n+1)\right] n^2$

$O\left[\frac{n^2 + 3n + 2}{6} \times n^2\right]$

$O(n^4)$

Q9.

```
void func^n (int n)
{ for (i = 1 to n) .
  {   for (j = 1 ; j ≤ n ; j = j + 1)
          print ("*") ;
              n^2 ;
      }
  }
```

$O(n + n^2 + n^2 + n^2)$

$O(3n^2 + n)$

$O(n^2)$