no 1.

Asymptotic Notation is a mathematic notation that is used to describe behaviour of a algorithm on basis of no. of inputs for eg if a ~~f~~ algorithm is represented as $f(n)$ when n is a no. of input asymptotic notation $O(n)$ for $f(n)$ tells that upper bound for $f(n)$ is n

that is $f(n) \leq c_1 n$

for $n > n_0, c_1 > 0$

Big(O) – Big O notation gives asymptotic upper bound of an algorithm for eg if a algorithm is represented ~~of~~ as $f(n)$ where n is no. of input $O(g(n))$ for $f(n)$ tells us

$$f(n) \leq c_1 g(n)$$

for $n > n_0$ $c_1 > 0$

Omega (Ω) – Omega notation gives asymptotic lower bound of an algorithm for eg if a algorithm is represented as $f(n)$ where n is no. of input $\Omega(g(n))$ for $f(n)$ tells us

$$c_1 g(n) \leq f(n)$$

for $n > n_0, c_1 > 0$

theta (Θ) theta notation gives asymptotic upper as well as lower bound of an algorithm for eg if a algorithm is represented as $f(n)$ where n is no. of inputs $\Theta(g(n))$ for $f(n)$ tells us

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

for $n > n_0$, $c_1 > 0$, $c_2 > 0$.

Ans 2.

for(i=1 ; i<=n ; i++) → (n+1) times

{ i=i*2; → (n) times

}

n + n + 1

= 2n + 1 (times)

time complexity = O(n)

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

$T(n) = 9T(n-2)$

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

$T(n) = 3^n T(0)$ $\quad T(0)=1$

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

Ans 4. $T(n) = 2T(n-1) - 1$

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

$T(n) = 4T(n-2) - \cancel{1} - 2$

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

$T(n) = 8T(n-3) - 1 - 2 - 4$

$T(n) = 2^n T(n-n) - 1 - 2 - 4 - 8 \text{ ---- } - 2^{n-1}$

acc to g p series

$a_n = a r^{n-1}$

$2^{n-1} = 1 \cdot r^{k-1}$

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

$k = n$

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

$= k = n$

Sum of series

$\frac{a(r^n - 1)}{r-1}$

$\frac{1(2^n - 1)}{2-1}$

$= 2^n - 1$

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

$T(n) = 1$

Ans 5.

Series

$1+2+3+4+5+6+7+8 \ldots k = n$

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

$$k(k+1) = 2n$$

$$k^2+k-2n = 0$$

$$k = \frac{-1+\sqrt{1-4n^2}}{2}$$

while (s<=n) k

{

i++ k-1

s = s+i; k-1

~~printf ("#")~~ k-1

}

$4k-1$

$$4 \times \frac{-1+\sqrt{1-4n^2}}{2}$$

$2\sqrt{1-4n^2}$

$= O(n)$

Ans 6.

$k^2 = n$

$k = \sqrt{n}$

for (i=1; i*i<=n; i++) $\sqrt{n}+1$

{

count++; $\sqrt{n}$

}

$O(\sqrt{n})$ $2\sqrt{n}+1$

7.

1 2 4 8 16 ... 2^K

$2^K = m$

$K = \log_m 2$

total $\log_m 2 + 1$

for (i=m/2; i<=m; i++) m/2

for (j=1; j<=m; j=j*2) $(\log_m 2 + 2) \times m/2$

for (K=1; K<=m; K=K*2) $(\log_m 2 + 2)^2 \times m/2$

count++ $(\log_m 2 + 1)^2 \times \frac{m}{2}$

$m \times (\log_m 2 + 2)^2$

$O(m(\log_m 2)^2)$

8.

~~T(m) = m^2 + m + 2~~

function (int m)

{ if (m==1) return;

for (i=1 to m) { $(m+1)$

for (j=1 to m) { $(m+1) \times (m+1)$

printf ("*"); $(m+1) \times (m+1)$

}

}

function (m-3)

$T(m) = 2m^2 + 4m + 2 + m + 1 + T(m-3)$

$\frac{m-2}{3} = K$

$T(m) = 2m^2 + 5m + 3 + T(m-3)$

$m = 1 + (K-1)3$

$m = 3K+2$

$T(m) = 2m^2 + 5m + 3 + 2(m-3)^2 + 5(m-3) + 3 + T(m-6)$

1 3 6 9 12 = m

$= 2m^2 + 2(m-3)^2 + 2(m-6)^2 + \dots 2(m-\frac{3m}{m})^2$

$+ 5(m) + 5(m-3) + 5(m-6) + \dots + T(m-3^m)$

Ans 8

~~T(1) = 1~~

~~T(2) = 4 + 16 + 3 + 1~~

~~= 18~~

~~T(3)~~

$= 2n^2 + 2(n-3)^2 + 2(n-6)^2 + 2(n-9)^2 - - - 2(n-3n)$
$+ 5(n) + 5(n-3) + 5(n-6) - - - 5(\frac{n}{3} - 3)2$
$+ 3\frac{(n-2)}{3}$

$O(3^{2n})$

Ans 9.

```
for (i=1 to n)
    for (j=1; j<=n; j=j+1)
    {
        print (*)
    }
```

$j \quad 1 + k + 1 \quad 2k+1 + 3k+3 \quad - - n$

t terms

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

$n = kt - k + 1$

$\frac{n+k-1}{k} = t$

now $k = i$

$n \times \frac{(n+n-1)}{n}$ + upper limit $n = \frac{n+n-1}{n}$
$= 2$
lower lim 1 $= \frac{n+1-1}{1} = n$

$n + n+1-1 \quad n+2 + n-3 - 2n$

$O(n)$ $= \frac{3n^2}{2}$

10. $n^k \geq c^n$

Omega is notation is relation

$n_0 > 1$

$c < n$

1 2 4 8 16 ... 2^k

$2^k = n$

$k = \log_2 n$

total

10. $n^k \geq c \cdot n$

Omega is notation is relation

$n_0 > 1$

$c < n$