Time Complexity Assignment

Ques 1)

given code →

```
for( int i = 1; i <= n; i++) {
    for( int j = 1; j <= i; j++)
    {
        sum ++;
    }
}
```

So the time complexity will be → $O(n^2)$.

Ques 2) Find $T(2)$, given $T(n) = 3T(n-1) + 12n$

$T(0) = 5$

$T(1) = 3T(0) + 12$

$= 15 + 12$

$= 27$

2
27
×3
81

$T(2) = 3T(1) + 24$

$= (27 \times 3) + 24$

$\Rightarrow 81 + 24 = \underline{105}$ (Ans)

Ques 3) Solve $T(n) = T(n-1) + C$ using substitution

$T(n) = T(n-1) + C$

$= (T(n-2) + C) + C$

$= T(n-2) + 2C$

Similarly $= T(n-3) + 3C$

Repeating this n times

So for $T(0) = T(0) + nC$

$=$

Date ...../...../.....

Page No. .....

given $T(0)$, you get a final solution. The time complexity is $O(n)$.

Ques 4)

Ans) Time complexity using masters theorem

given $T(n) = 16T(n/4) + n^2 \log n$

Comparing with standard form → $T(n) = aT(n/b) + f(n)$

$a = 16$

$b = 4$

$f(n) = n^2 \log n$

$\log_b^a = \log_4^{16} = 2$

Since $f(n) = n^2 \log n$, grows faster than n^2, the time complexity is dominated by $f(n)$. So by Master's theorem the complexity is

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

Ques 5) Assume we have for $T(n) = 2T(n/2) + n$

- At the root, work done is n
- At the next level, the work is $2(n/2) = n$
- At the next level, the work is $4(n/4) = n$

Since each level does work of n, and the height of the tree is $(\log n)$ the total work is $(n \times \log n)$

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

Teacher's Sign

Ques 6) As in question 5, the tree structure is same, but now the additional work at each level is a constant k. The total work is →

$$k \cdot \log n$$

So the time complexity will be →

$$O(k \log n)$$