Eg:- Question 1: Time Complexity Analysis and Improvement

```
int sum = 0;
for(int i = 1; i < n; i++) {
    for(int j = 1; j <= i; j++) {
    }
    sum++;
}
```

Eg:- Question 2: Find the Value of (T(2))

Recurrence Relation:

[T(n) = 3T(n-1) + 12n]
[T(0) = 5]

// We need to find (T(2)). First, we compute (T(1)):

[T(1) = 3T(0) + 12 cdot 1]
[T(1) = 3 cdot 5 + 12 = 15 + 12 = 27]

Now compute (T(2)):

[T(2) = 3T(1) + 12 cdot 2]
[T(2) = 3 cdot 27 + 24 = 81 + 24 = 105]

Question 3: Solve the Recurrence Relation (T(n) = T(n-1) + c)

Substitution Method:

1. Base Case:
   Assume (T(0) = T_0) (constant).

2. Unwind the Recurrence:

   T(n) = T(n-1) + c

   T(n-1) = T(n-2) + c

   T(n-2) = T(n-3) + c

   T(1) = T(0) + c
   Adding these up:

$T(n) = T(0) + c.n$

$T(n) = T_0 + cn$

So the solution is:

$T(n) = O(n)$

Eg:- Question 4: Time Complexity Using the Master Theorem

Recurrence Relation:

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

Master Theorem:
[$T(n) = aT(n/b) + f(n)$]

Where:
- ($a = 16$)
- ($b = 4$)
- ($f(n) = n^2 \log n$)

Master Theorem Cases:

1. Compute ($\log_b a$):
   [
   $\log_b a = \log_4 16 = \log_4 (4^2) = 2$
   ]

2. Compare ($f(n)$) to ($n^{\log_b a}$):
   [
   $f(n) = n^2 \log n$
   ]
   [
   $n^{\log_b a} = n^2$
   ]

   Since ($f(n) = n^2 \log n$) grows faster than (n^2), we are in Case 3 of the Master Theorem where ($f(n) = \Omega(n^c)$) for ($c = 2$) and ($f(n)$) is polynomially larger.

Thus:
[
T(n) = Theta(f(n)) = Theta(n^2 log n)
]

Eg:- Question 5: Solve Recurrence Relation Using Recursion Tree Method

Recurrence Relation:

[T(n) = 2T(n/2) + n]

Recursion Tree Method:

1. Draw the Recursion Tree:

   - At level 0: (T(n))
   - At level 1: (2 times T(n/2))
   - At level 2: (2^2 times T(n/4))
   - At level (k): (2^k times T(n/2^k))

2. Cost at Each Level:

   - Level 0: (n)
   - Level 1: (2 times frac{n}{2} = n)
   - Level 2: (2^2 times frac{n}{4} = n)
   - The cost at each level is (n).

3. Number of Levels:

   The recursion tree has (log_2 n) levels.

4. Total Cost:

   [
   text{Total Cost} = n times log_2 n = Theta(n log n)
   ]

Eg:- Question 6: Solve Recurrence Relation Using Recursion Tree Method

Recurrence Relation:

[T(n) = 2T(n/2) + K]

Recursion Tree Method:

1. Draw the Recursion Tree:

   - At level 0: (T(n))
   - At level 1: (2 times T(n/2))
   - At level 2: (2^2 times T(n/4))
   - At level (k): (2^k times T(n/2^k))

2. Cost at Each Level:

   - Level 0: (K)
   - Level 1: (2 times K = 2K)
   - Level 2: (2^2 times K = 4K)
   - The cost at level (k) is (2^k times K = K times 2^k).

3. Number of Levels:

   The recursion tree has (log_2 n) levels.

4. Total Cost:

   The total cost is the sum of costs at all levels:

   [
   text{Total Cost} = K times (1 + 2 + 4 + cdots + 2^{log_2 n}) = K times (2^{log_2 n} - 1) = K times (n - 1) = Theta(n)
   ]

   So the time complexity is (Theta(n)).