# Solution of Mini-Assignment

**Question 1**

```
for(int i = 1;i<=n;i++)
{
        for(int j = n;j>=1;j–)
        {
                System.out.println(“CSIT”);
        }
}
```

Here inner loop complexity is **n** because the inner loop runs n times for any value of n.
Here the outer loop also runs n times.
So, at iteration i = 1, j loop runs n times
So, at iteration i = 2, j loop runs n times
.
.
.
So, at iteration i = n, j loop runs n times, so tidal time is

n+n+..... n times
n*n
n^2
So maximum power of n is 2

**T(n) = O(n^2)**

**Question 2**

```
for(int i = 1;i<=n;i++)
{
        System.out.println(“CSIT”);
}
for(int j = n;j>=1;j–)
{
        System.out.println(“CSIT”);
}
```

Sankar Kumar Mridha

Here the first loop complexity is **n** because the loop runs n times for any value of n. Similarly, the complexity of the second loop is **n** because the loop runs n times for any value of n.

So total time is
n+n
2n
So maximum power of n is 1

**T(n) = O(n)**

**Question 3**

```
for(int i = 1;i<=n;i++)
{
        System.out.println(“CSIT”);
}
for(int i = 1;i<=n;i++)
{
        for(int j = 1;j<=m;j = j*2)
        {
                System.out.println(“CSIT”);
        }
 }
```

Here the first loop complexity is **n** because the loop runs n times for any value of n. Similarly, the complexity of the second loop is **n** because the loop runs n times for any value of n. But the inner loop runs $2^{i-1}$ time
At iteration 1, j loop runs 1 times = $2^{1-1}$
At iteration 2, j loop runs 2 times = $2^{2-1}$
At iteration 3, j loop runs 3 times = $2^{3-1}$
.
.
.
At iteration k, j loop runs $2^{k-1}$
As per the statement
$2^{k-1} = n$
$(k-1)\log_2 2 = \log_2 n$

Sankar Kumar Mridha

$K = \log_2 n + 1$
Total complexity = $n + n*(\log_2 n + 1) = 2n + n\log_2 n$

So maximum power of n is $\log_2 n$

**$T(n) = O(n\log_2 n)$**

**Question 4**

```
for (i = 0; i < N; i++)
{
        a = a + rand();
}
for (j = 0; j < M; j++)
{
        b = b + rand();
}
```

Here the first loop complexity is **N** because the loop runs N times for any value of N.
Similarly, the complexity of the second loop is **M** because the loop runs M times for any value of M.
So total time is
N+M

Here the value of N and M is user input. So

**$T(n) = O(N+M)$**

**Question 5**

```
for (i = 0; i < N; i++)
{
        for (j = N; j > i; j--)
        {
                a = a + i + j;
        }
}
```

Here the outer loop complexity is **N** because the loop runs N (0 to N-1) times for any value of N.
But inner loop runs based on the value of i in outer loop

Sankar Kumar Mridha

when  i = 0 , j runs N times
when  i = 1 , j runs N -1 times
.
.
.
When i = n-1, j runs 1 times
So total time is
N + (N-1) + (N-2) + …. + 2 + 1
(N(N+1))/2
N^2/2 + N/2  maximum power on N is 2

**T(n) = O(N^2)**

**Question 6**

```
for(int i = 1;i<=n;i=i*5)
{
        System.out.println("CSIT");
}
```

Here, the inner loop runs 5^$^{i-1}$ time
At iteration 1, j loop runs  1 times = 5^$^{1-1}$
At iteration 2, j loop runs  5 times = 5^$^{2-1}$
At iteration 3, j loop runs  25 times = 5^$^{3-1}$
.
.
.
At iteration k, j loop runs   5^$^{k-1}$
As per the statement
5^$^{k-1\ =}$= n
$(k-1)\log_5 5 = \log_5 n$
$K = \log_5 n + 1$

So maximum power of n is $\log_2 n$

**T(n) = O($\log_5$n)**

Sankar Kumar Mridha

**Question 7**

```
for(int i = 1;i<=n;i++)
{
        for(int j = 1;j<=m;j = j*2)
        {
                for(k=j;k>=1;k–)
                {
                        System.out.println(“CSIT”);
                }
        }
}
```

Here the outer loop runs n time.
First inner loop runs $\log_2 m$ times.
Inner loop also runs $\log_2 m$ times.
Total complexity $n*\log_2 m*\log_2 m$

$$T(n) = O(n*\log_2 m*\log_2 m) \text{ or } O(n*\log_2 m^2)$$

**Question 8**

```
for(int j = 1; j < i; j *= 2)
{
        for(int k = j; k >= 1; k /= 2)
        {
                System.out.println(“DSA”);
        }
}
```

Outer loop runs 2^i-1 time
At iteration 1, j loop runs 1 times = $2^{\wedge 1-1}$
At iteration 2, j loop runs 2 times = $2^{\wedge 2-1}$
At iteration 3, j loop runs 4 times = $2^{\wedge 3-1}$
.
.
.
At iteration p, j loop runs $2^{\wedge p-1}$
As per the statement

Sankar Kumar Mridha

$2^{\wedge p-1} = i$
$(p-1)\log_2 2 = \log_2 i$
$p = \log_2 i + 1$

Inner loop depends on the value of j variable
At iteration 1, j loop runs 2

Outer loop runs 2^i-1 time
At iteration 1, j loop runs 1 times = $2^{\wedge 1-1}$
At iteration 2, j loop runs 2 times = $2^{\wedge 2-1}$
At iteration 3, j loop runs 4 times = $2^{\wedge 3-1}$
.
.
.
At iteration k, j loop runs $2^{\wedge k-1}$
As per the statement
$2^{\wedge k-1} = n$
$(k-1)\log_2 2 = \log_2 n$
$k = \log_2 n + 1$

Total complexity
p*k
$(\log_2 i + 1)*(\log_2 n + 1)$
$= \log_2 n * \log_2 i + \log_2 n + \log_2 i + 1$

**$T(n) = O(\log_2 i * \log_2 n)$**

Sankar Kumar Mridha