Page No.
Date

Problem 5

Dry Run & Analyze: Time and space Complexity.

1. Dry run the code for n = 4. How many times is printed? what is the time complexity?

```
Void printTriangle(int n){
    for (int i = 0; i<n; i++)
        for (int j=0; j<=i; j++)
            System.out.print (" *");
}.
```

Ans:-
Dry Run for n = 4:
n = 4
- i = 0 : j runs 0 → prints 1*
- i = 1 : j = 0,1 → prints 2*
- i = 2 : j = 0,1,2 → prints 3*
- i = 3 : j = 0,1,2,3 → prints 4*

Total * printed 1+2+3+4 = 10

Time complexity
- outer loop runs n times.
- Inner loop runs i+1 times.
- Total iterations : $1+2+\dots\dots+n = n(n+1)/2$
$= O(n^2)$

- Total * : 10
- Time complexity : $O(n^2)$
- Space complexity : $O(1)$

Page No.
Date

Q2 Dry run for n =8. what's the number of iterations ? Time Complexity.

```
void printPattern(int n){
    for(int i=1; i<=n; i* =2)
        for(int j=0; j<n; j++)
            System.out.println(i+","+j);
}
```

Ans:- Dry run for n =8:

- i : 1, 2, 4, 8 $\longrightarrow$ 4 iterations
- inner loop runs n = 8 times for each i

Total iterations = 4×8 = 32

Time Complexity
- outer loop: O(log n)
- inner loop: O(n)
- Total: O(n log n).

Answer:-
Total iterations: 32
Time Complexity: O(n log n)
Space Complexity: O(1)

Page No. Date

Q3. Dry run for n =20. How many recursive calls ? what values are printed?

```
void recHalf(int n) {
    if (n <=0) return;
        System.out.print(n+" ");
        recHalf (n/2);
}
```

Ans- Dry run for n = 20:

- Call recHalf (20) → prints 20
- recHalf (10) → prints 10
- recHalf (5) → prints 5
- recHalf (2) → prints 2
- recHalf (1) → prints 1
- recHalf (0) → stops

Printed values : 20 10 5 2 1
Recursive calls 6 (including base case)

Time complexity : O(log n)
- Each call divides n by 2

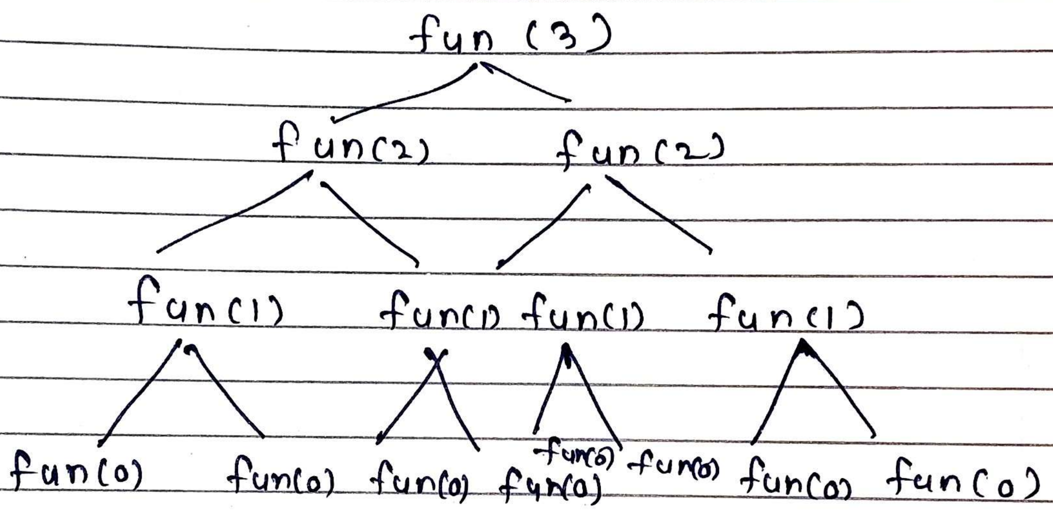
Answer :-
- printed : 20 10 5 2 1
- Recursive calls : 6
- Time complexity : O(log n)
- space complexity : O(log n)

Page No.
Date

4. Dry run for n=3. How many total calls are made? what's the time complexity?

```
void fun(int n) {
    if (n ==0) return;
    fun(n-1);
    fun(n-1);
}
```

Ans3



Total calls.
This form a full binary tree with $2^n - 1$ calls For $n=3$, total $2^3 - 1 = 7$

Time complexity : $O(2^n)$
Space complexity : $O(n)$

Answer.
Total calls : 7
Time complexity : $O(2^n n)$
Space complexity : $O(n)$

Page No.
Date

5. Dry run for n = 3. How many total iterations?
Time Complexity?

```
void tripleNested (int n) {
    for (int i=0; i<n; i++)
        for(intj =0; j<n; j++)
            for (int k=0; k<n; k++)
                System.out.println(i+j+k);
}
```

Ans:- Dry run for n=3
- All three loops run from 0 to 2
- Total iterations :- $n \times n \times n = 3^3 = 27$.

Time complexity : $O(n^3)$
- Each loop runs n times

Answer
- Total iterations : 27
- Time Complexity : $O(n^2)$
- Space Complexity : $O(1)$