Array Printing Using Recursion - Quick Revision Notes (Hinglish)

Code ka Logic:

- Function `print(int n, int j, int arr[])` recursion ka use karke ek array ke elements print karta hai.
- **Base Condition**:
  - Agar `j == n`, toh function `n+1` return karta hai aur recursion stop ho jata hai.
- **Recursive Calls**:
  - Pehle `arr[j]` print hota hai.
  - Fir function `print(n, j+1, arr)` call hota hai, jo agle element pe move karta hai.
- Is tarah, array ke sare elements sequence mein print ho jate hain.

Problem Example:

Problem Statement:

Ek integer array diya gaya hai, jisme recursion ka use karke sare elements print karne hain.

Example:

Input: arr[] = {10, 20, 30, 40, 50}, n = 5

Output: 10 20 30 40 50

```
#include<iostream>
using namespace std;

// Recursion ka use karke array ke elements print karne ka function
int print(int n, int j, int arr[]) {
    if(j == n) return n+1; // Base case recursion stop karne ke liye

    cout << arr[j] << " "; // Current element print karo
    print(n, j+1, arr); // Next element ke liye recursive call
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    cout << print(4, 0, arr); // Function call array elements print karne ke liye
}
```

Dry Run of the Code (For print(4,0,arr))

Function Call	j	Output
print(4, 0, arr)	0	1
print(4, 1, arr)	1	2
print(4, 2, arr)	2	3
print(4, 3, arr)	3	4
print(4, 4, arr)	4	5
print(4, 5, arr)	5	Returns 5

Final Output for print(4,0,arr):

1 2 3 4 5 5

Time Complexity:

- Har function call ek element process karta hai.
- Function `n+1` baar call hota hai (base case ko include karke).
- Isliye, time complexity **O(n)** hai.

Key Takeaways:

- Recursion ka use array ko iterate karne ke liye ho sakta hai.
- Base case recursion stop karne ke liye zaroori hai.
- Ye approach array ke elements ko order-wise print karti hai.