Array Printing Using Recursion - Quick Revision Notes

### Logic of the Given Code:

- The function `print(int n, int j, int arr[])` prints elements of an array recursively.
- **Base Condition**:
  - If `j == n`, the function returns `n+1` to terminate recursion.
- **Recursive Calls**:
  - Print `arr[j]` before making the recursive call.
  - Call `print(n, j+1, arr)`, which moves to the next index.
- This ensures that all elements of the array are printed sequentially.

Problem Example:

Problem Statement:

Given an array of integers, print all elements of the array using recursion.

Example:

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

Output: 10 20 30 40 50

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

// Function to print elements of an array using recursion
int print(int n, int j, int arr[]) {
    if(j == n) return n+1; // Base condition to stop recursion

    cout << arr[j] << " "; // Print the current element
    print(n, j+1, arr); // Recursive call for the next element
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    cout << print(4, 0, arr); // Calling the function to print elements
}
```

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:

- Each function call processes one element of the array.
- The function is called `n+1` times (including the base case).
- Thus, the time complexity is **O(n)**.

Key Takeaways:

- Recursion can be used to iterate over an array.
- A base condition is necessary to stop infinite recursion.
- This approach prints the array elements in order.