Q. What happens when we print the name of the array in C or C++?

int arr[ ] = { 2, 3, 6, 4, 9 }; (addresses: 100, 104, 108, 112, 116)

Pointer

printf("%d", arr); = 100

It prints the address of the 1st element of the array. That means, the name of the array is actually a pointer → "Array Pointer".
Arrays are directly "Call by Reference".

3 4 6 2 1 0 — **MCQ**

```
for ( int i = 0 ; i < size; i++ )
{
    printf("%d", *arr + i)   → 3, 4, 5, 6, 7, 8
    printf("%d", (*arr)+i)   → 3, 4, 5, 6, 7, 8
    printf("%d", (*arr+i))   → 3, 4, 5, 6, 7, 8
    printf("%d", *(arr+i))   → 3, 4, 6, 2, 1, 0
}
```

DRY RUN: →

res = 0

0000 0
0001 1
———
0001
0010 2
———
0011
0011 3
———
0000
0100 4
———
0100
0011 3
———
0111
0010 2
———
0101
0001 1
———
0100 → 4

1, 2, 3, 4, 3, 2, 1 (4 → 7)

Swap two variables without using third variable.

x = 15
y = 21

x = x ^ y
y = x ^ y = 15
x = x ^ y = 21

x 15 01111
y 21 10101
x = 11010 = 26
y = 10101
x^y 01111 = 15

11010
01111
———
10101 = 21

x = 15
y = 21

x = x+y
y = x-y
x = x-y

x = x+y
= 15+21
= 36
y = x-y
= 36 − 21
= 15
x = x-y = 36-15
= 21 //

C Language: → (DMA) → Pointers
4 functions → #include <stdlib.h>

garbage values ← (I) malloc → memory allocation [garbage]
void
default ← (II) calloc → continuous/contiguous memory allocation
(III) realloc → re-allocation
(IV) free → deallocation

10 blocks = 4 bytes — 40 bytes — int → 4 bytes (Type Casting)

Syntax: <stdlib.h>

malloc (size);
calloc(numberofelements, size);
realloc (memory/ptr ; new size);
free (memory/ptr);

C++ ← Object Oriented Programming
C + SIMULA = C++ → Bjarne Stroustrup
SIMULA ↓ Language of flowcharts

C + C++ = Java → James Gosling

Python 1991 or Java 1995
Guido van Rossum

DSA Roadmap for ECE/EIE/Robotics: →
Basics of C (DMA in C)
* Basics of C++ + OOPs (DMA in C++)
Arrays → Indepth + Questions
Algorithms → Searching & Sorting Algorithms — Time Complexity
Data Structures →
→ Linear Data Structures
→ Non Linear Data Structures
Next → Recursion, Backtracking, Greedy Algos, Dynamic Programming
Bit Masking,