Revision Notes: Find Two Unique Numbers Using XOR

Problem Statement:

Given an array where every number appears exactly twice except for two numbers that appear only once, find those two unique numbers using Bitwise XOR.

Key Properties of XOR:

- a ^ a = 0 (same numbers cancel out)
- a ^ 0 = a
- XOR is commutative and associative

Algorithm Steps:

- Step 1: XOR all elements of the array to get res = unique1 ^ unique2.
- Step 2: Find the first set bit position k in res.
- Step 3: Divide the array elements based on the k-th bit.
- Step 4: XOR the selected group to get the first unique number.
- Step 5: XOR this value with res to get the second unique number.

Time & Space Complexity:

Time Complexity: O(n)
Space Complexity: O(1)

Important Interview Line:

First XOR all elements to cancel duplicates, then use the rightmost set bit to separate the two unique numbers.

Code with Clear Comments:

```
void findUniques(int *arr, int n)
{
int res = 0;

// Step 1: XOR all elements
for(int i = 0; i < n; i++)
res ^= arr[i];

// Step 2: Find first set bit in res
int temp = res, k = 0;
while((temp & 1) == 0)
{
temp >>= 1;
k++;
```

```
}

// Step 3: XOR elements having k-th bit set
int firstUnique = 0;
for(int i = 0; i < n; i++)
if((arr[i] >> k) & 1)
firstUnique ^= arr[i];

// Step 4: Find second unique number
int secondUnique = firstUnique ^ res;

cout << firstUnique << " " << secondUnique;
}
```