Data Structure and Operations: **Day-12 (Bit Manipulation, : Unique Elements Identification):**

Task 1: Bit Manipulation Basics

Create a function that counts the number of set bits (1s) in the binary representation of an integer. Extend this to count the total number of set bits in all integers from 1 to n.

```java
package wipro.com.assignment09;
public class BitManipulation {
    // Function to count the number of set bits in an integer
    public static int countSetBits(int n) {
        int count = 0;
        while (n > 0) {
            count += n & 1;
            n >>= 1;
        }
        return count;
    }
    // Function to count the total number of set bits in all integers from 1 to n
    public static int totalSetBits(int n) {
        int total = 0;
        for (int i = 1; i <= n; i++) {
            total += countSetBits(i);
        }
        return total;
    }

    public static void main(String[] args) {
        int number = 13;  // Example integer
        System.out.println("Number of set bits in " + number + " is: " +
countSetBits(number));

        int range = 5;  // Example range
        System.out.println("Total number of set bits from 1 to " + range + " is: " +
totalSetBits(range));
    }
}
```

Output:

```
Number of set bits in 13 is: 3
Total number of set bits from 1 to 5 is: 7
```

Task 2: Unique Elements Identification Given an array of integers where every element appears twice except for two, write a function that efficiently finds these two non-repeating elements using bitwise XOR operations.

```
package wipro.com.assignment09;
public class UniqueElements {
    // Function to find the two non-repeating elements in an array
    public static int[] findUniqueElements(int[] nums) {
        // Step 1: XOR all the elements in the array
        int xor = 0;
        for (int num : nums) {
            xor ^= num;
        }

        // Step 2:Find a set bit in the result (this differentiates the two unique
numbers)
        int setBit = xor & -xor;

        // Step 3: Divide the elements into two groups and XOR them separately
        int unique1 = 0, unique2 = 0;
        for (int num : nums) {
            if ((num & setBit) == 0) {
                unique1 ^= num;
            } else {
                unique2 ^= num;
            }
        }

        // Return the two unique elements
        return new int[] {unique1, unique2};
    }

    public static void main(String[] args) {
        int[] nums = {4, 1, 2, 1, 2, 3, 4, 5};  // Example array
        int[] result = findUniqueElements(nums);
        System.out.println("The two non-repeating elements are: " + result[0] + " and
" + result[1]);
    }
}
```

Output:

```
The two non-repeating elements are: 5 and 3
```