Name- Amit Kumar Parhi

Email- amitkparhi07@gmail.com

Day 12:

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.

Solution:

```
package com.wipro;

public class BitManipulation {

    // Function to count the number of set bits in the binary representation of an
integer
    public static int countSetBits(int num) {
        int count = 0;
        while (num > 0) {
            count += num & 1;
            num >>= 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 num = 13;

System.out.println("Number of set bits in " + num + " is: " + countSetBits(num));


int n = 5;

System.out.println("Total number of set bits from 1 to " + n + " is: " +
totalSetBits(n));
    }
}
```

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.

Solution:

```
package com.wipro;

public class UniqueElements {

    public static int[] findUniqueElements(int[] nums) {
        // Step 1: XOR all elements to get the XOR of the two unique numbers
        int xorResult = 0;
        for (int num : nums) {
            xorResult ^= num;
        }

        // Step 2: Find a bit that is set in the xorResult (rightmost set bit)
        int setBit = xorResult & ~(xorResult - 1);

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

        return new int[]{unique1, unique2};
    }

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

Output:

The two unique elements are: 3 and 5