Himanshu Kamane
kamanehimanshu76@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.

Code

```
package WiproEP;
public class BitCounter {
    public static void main(String[] args) {
        int n = 10; // Change this to the desired number
        System.out.println("Total number of set bits from 1 to " + n + ": " + countSetBits(n));
    }
    public static int countSetBits(int n) {
        int totalSetBits = 0;
        for (int i = 1; i <= n; i++) {
            totalSetBits += countSetBitsInNumber(i);
        }
        return totalSetBits;
    }
    public static int countSetBitsInNumber(int n) {
        int setBits = 0;
        while (n > 0) {
            setBits += n & 1;
            n >>= 1;
        }
        return setBits;
    }
}
```

Output

```
Total number of set bits from 1 to 10: 17
```

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.

Code

```
package WiproEP;
	public class UniqueElements {
		public static int[] findNonRepeatingElements(int[] arr) {
			int xorResult = 0;
			for (int num : arr) {
				xorResult ^= num;
			}
			// Finding the rightmost set bit
			int rightmostSetBit = xorResult & -xorResult;
			int[] result = new int[2];
			// Splitting the array into two groups based on the rightmost set bit
			for (int num : arr) {
				if ((num & rightmostSetBit) != 0) {
					result[0] ^= num;
				} else {
					result[1] ^= num;
				}
			}
			return result;
		}
		public static void main(String[] args) {
			int[] arr = {2, 4, 6, 8, 10, 2, 4, 6, 9, 8};
			int[] result = findNonRepeatingElements(arr);
			System.out.println("Non-repeating elements: " + result[0] + ", " + result[1]);
		}
	}
```

Output

```
Non-repeating elements: 9, 10
```