DAY 4

BIT WISE OPERATOR PROBLEMS:

1. **Swap two numbers:**

```
public class BitwiseSwap {
    public static void main(String[] args) {
        int a = 5, b = 10;

        System.out.println("Before swapping: a = " + a + ", b = " + b);



        a = a ^ b;
        b = a ^ b;
        a = a ^ b;

        System.out.println("After swapping: a = " + a + ", b = " + b);
    }
}
```

2. **Check if a Number is Even or Odd:**

Logic: A number is **even** if its last bit is 0 and **odd** if its last bit is 1.
Use n & 1: If n & 1 is 0, it's even; otherwise, it's odd.

```
public class EvenOddCheck {
    public static void main(String[] args) {
        int n = 7;
       if ((n & 1) == 0)
            System.out.println(n + " is Even");
        else
            System.out.println(n + " is Odd");
    }
}
```

3. **Count the Number of Set Bits (1s) in a Number:**

Logic: Use n & (n - 1), which removes the rightmost set bit.

```
public class CountSetBits {
    public static int countBits(int n) {
        int count = 0;
        while (n > 0) {
            n = n & (n - 1);
            count++;
        }
        return count;
    }

    public static void main(String[] args) {
        int n = 29;
        System.out.println("Set Bits: " + countBits(n));
    }
}
```

4. **Find the Missing Number in a Range (Using XOR):**

Logic: XOR all numbers from 1 to n and XOR all array elements, then XOR both results.

```
public class MissingNumber {
    public static int findMissing(int[] arr, int n) {
        int xorAll = 0, xorArr = 0;

        for (int i = 1; i <= n; i++) xorAll ^= i;
        for (int num : arr) xorArr ^= num;

        return xorAll ^ xorArr;
    }

    public static void main(String[] args) {
        int[] arr = {1, 2, 4, 5, 6};
        System.out.println("Missing Number: " + findMissing(arr, 6));
    }
}
```

5. **Reverse Bits of a Number:**

Logic: Extract bits one by one and shift them in reverse order.

```
public class ReverseBits {
    public static int reverseBits(int n) {
        int result = 0;
        for (int i = 0; i < 32; i++) {
            result = (result << 1) | (n & 1);
            n >>= 1;
        }
        return result;
    }

    public static void main(String[] args) {
        int n = 5; // Binary: 000...101 -> Reverse: 101...000
        System.out.println("Reversed Bits: " + reverseBits(n));
    }
}
```

JAVA CODE ILLUSTRATING OOPS CONCEPTS:

```
interface Payroll {
    double calculateSalary();
}

abstract class Employee implements Payroll {
    private String name;
    private int id;
    protected double baseSalary;
    public Employee(String name, int id, double baseSalary) {
        this.name = name;
        this.id = id;
        this.baseSalary = baseSalary;
    }
    public String getName() {
        return name;
    }

    public int getId() {
        return id;
    }

    public double getBaseSalary() {
        return baseSalary;
```

```
    }

    public abstract double calculateSalary();

        public void displayInfo() {
        System.out.println("Employee ID: " + id);
        System.out.println("Name: " + name);
        System.out.println("Salary: " + calculateSalary());
    }
}

class FullTimeEmployee extends Employee {
    private double bonus;
    public FullTimeEmployee(String name, int id, double baseSalary, double bonus) {
        super(name, id, baseSalary);
        this.bonus = bonus;
    }
    @Override
    public double calculateSalary() {
        return baseSalary + bonus;
    }
}

class PartTimeEmployee extends Employee {
    private int hoursWorked;
```

```
    private double hourlyRate;

    public PartTimeEmployee(String name, int id, double hourlyRate, int hoursWorked) {
        super(name, id, 0);
        this.hourlyRate = hourlyRate;
        this.hoursWorked = hoursWorked;
    }
    @Override
    public double calculateSalary() {
        return hoursWorked * hourlyRate;    }
}
public class HRManagementSystem {
    public static void main(String[] args) {
        FullTimeEmployee emp1 = new FullTimeEmployee("Alice", 101, 5000, 1000);
        emp1.displayInfo();
        System.out.println("------------------------------");
              PartTimeEmployee emp2 = new PartTimeEmployee("Bob", 102, 20, 80);
        emp2.displayInfo();
    }
}
```