DAY 8 :

ASSIGNMENT 1:

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.

ANSWER:

Step 1: Count the number of set bits in a single integer

We can use Brian Kernighan's Algorithm to count the number of set bits in an integer. The idea of the algorithm is that it repeatedly turns off the rightmost set bit of the number and counts how many times this operation can be performed.

java

```
public class BitManipulation {
    // Function to count the number of set bits in a single integer
    public static int countSetBits(int num) {
        int count = 0;
        while (num > 0) {
            num &= (num - 1); // turn off the rightmost set bit
            count++;
        }
        return count;
    }

    public static void main(String[] args) {
        int num = 29; // example number
        System.out.println("Number of set bits in " + num + " is: " + countSetBits(num));
    }
}
```

Step 2: Count the total number of set bits in all integers from 1 to n

To count the total number of set bits in all integers from 1 to \(n \), we can iterate through each number from 1 to \(n \) and use the countSetBits function.

java

```
public class BitManipulation {
    // Function to count the number of set bits in a single integer
    public static int countSetBits(int num) {
        int count = 0;
        while (num > 0) {
            num &= (num - 1); // turn off the rightmost set bit
            count++;
        }
        return count;
    }

    // Function to count the total number of set bits in all integers from 1 to n
    public static int countTotalSetBits(int n) {
        int totalSetBits = 0;
        for (int i = 1; i <= n; i++) {
            totalSetBits += countSetBits(i);
        }
        return totalSetBits;
    }

    public static void main(String[] args) {
        int n = 5; // example number
        System.out.println("Total number of set bits from 1 to " + n + " is: " + countTotalSetBits(n));
    }
}
```

Explanation

1. *countSetBits Function:*

   - This function uses a loop to turn off the rightmost set bit of the number using num &= (num - 1) and increments the count each time.

   - The loop continues until all bits are turned off.

2. *countTotalSetBits Function:*

   - This function iterates through all numbers from 1 to \(n \).

   - For each number, it calls countSetBits and adds the result to totalSetBits.

3. *main Method:*

   - The main method demonstrates the usage of both functions by counting the number of set bits in a single integer and the total number of set bits from 1 to \(n \).

This approach is simple and easy to understand, although not the most optimized for large values of \(n \). For large \(n \), more sophisticated bit manipulation techniques might be necessary to improve efficiency.