Day 22 Assignment

Name: Mehul Anjikhane Email: mehulanjikhane13@gmail.com

**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.**

**package** algorithms;

**public** **class** BitManipulation {

// Count set bits in an integer using Brian Kernighan's algorithm

**public** **static** **int** countSetBits(**int** n) {

**int** count = 0;

**while** (n > 0) {

// This operation resets the least significant set bit to 0

n &= (n - 1);

count++;

}

**return** count;

}

// Count set bits in all integers from 1 to n (naive approach)

**public** **static** **int** countSetBitsNaive(**int** n) {

**int** totalSetBits = 0;

**for** (**int** i = 1; i <= n; i++) {

// Call the previous function for each number

totalSetBits += *countSetBits*(i);

}

**return** totalSetBits;

}

// Convert Decimal to Binary

**public** **static** String decToBinary(**int** num) {

**if** (num == 0) {

**return** "0";

}

StringBuilder binary = **new** StringBuilder();

**while** (num > 0) {

**int** remainder = num % 2;

binary.append(remainder);

num /= 2;

}

**return** binary.reverse().toString();

}

**public** **static** **void** main(String[] args) {

**int** number = 15;

**int** setBits = *countSetBits*(number);

System.***out***.println("Number: " + number + ", Binary Representation: " + *decToBinary*(number));

System.***out***.println("Number of set bits in " + number + ": " + setBits);

**int** n = 5;

**int** totalSetBits = *countSetBitsNaive*(n); // Not efficient for large n

**for** (**int** i = 1; i <= n; i++) {

System.***out***.println("Number: " + i + ", Binary Form: " + *decToBinary*(i));

}

System.***out***.println("Total set bits from 1 to " + n + ": " + totalSetBits);

}

}

**Output:**

Number: 15, Binary Representation: 1111

Number of set bits in 15: 4

Number: 1, Binary Form: 1

Number: 2, Binary Form: 10

Number: 3, Binary Form: 11

Number: 4, Binary Form: 100

Number: 5, Binary Form: 101

Total set bits from 1 to 5: 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** algorithms;

**import** java.util.Arrays;

**public** **class** UniqueElementsIdentificationUsing\_BitwiseXOR {

// Find two unique elements using XOR

**public** **static** **void** findUniqueElements(**int**[] arr) {

**int** xor = 0;

**for** (**int** num : arr) {

xor ^= num;

}

// Get the rightmost set bit in XOR (separates numbers with different LSBs)

**int** rightmostSetBit = xor & ~(xor - 1);

**int** unique1 = 0, unique2 = 0;

**for** (**int** num : arr) {

// If the rightmost set bit of num and rightmostSetBit are the same, add num to unique1

**if** ((num & rightmostSetBit) != 0) {

unique1 ^= num;

} **else** {

unique2 ^= num;

}

}

System.***out***.println("Unique Elements: " + unique1 + ", " + unique2);

}

**public** **static** **void** main(String[] args) {

**int**[] arr = { 7, 3, 5, 4, 5, 3, 7, 1 };

System.***out***.println("Given Array: " + Arrays.*toString*(arr));

*findUniqueElements*(arr);

}

}

**Output:**

Given Array: [7, 3, 5, 4, 5, 3, 7, 1]

Unique Elements: 1, 4