Problem 1: given a number, print its binary representation. [easy]

Input 1: number = 5

Output 1: 101

Input 2: number = 10

Output 2: 1010

import java.util.Scanner;

public class fundNum{

    public static void main(String[] args) {

        StringBuilder str = new StringBuilder();

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the number : ");

        int num = sc.nextInt();

        while(num>0){

            int modulo = num%2;

            str.insert(0,modulo);

            num = num/2;

        }

        System.out.println(str.toString());

    }

}

Problem 2: given a number ‘n’, predict whether it is a power of two or not. [medium]

Input 1: n = 15

Output 1: False

Input 2: n = 32

Output 2: True

import java.util.Scanner;

public class fundNum{

    public static boolean powOfTwo(int num){

        if(num<=0){

            return false;

        }

        else{

            if((num & (num-1))==0){

                return true;

            }

        }

        return false;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int num = sc.nextInt();

        boolean power  = powOfTwo(num);

        if(!power)

        System.out.println("false");

        else{

            System.out.println("true");

        }

    }

}

Q3. Problem 1: Given a number. Using bit manipulation, check whether it is odd or even.

Input 8, Even

3, False

import java.util.Scanner;

public class fundNum{

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int num = sc.nextInt();

        if((num & 1) == 1){

            System.out.println("false");

        }

        else{

            System.out.println("even");

        }

    }

}

Q4. Given a number, count the number of set bits in that number without using an extra space.

Note : bit ‘1’ is also known as set bit.

import java.util.Scanner;

public class fundNum{

    public static int powOfTwo(int num){

        int count = 0;

        while(num>0){

            int calc = num & 1;

            count = count + calc;

            num >>= 1;

        }

        return count;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int num = sc.nextInt();

        powOfTwo(num);

        System.out.println(powOfTwo(num));

    }

}

Q5. Given an integer array, duplicates are present in it in a way that all duplicates appear an even

number of times except one which appears an odd number of times. Find that odd appearing

element in linear time and without using any extra memory.

For example,

Input : arr[] = [4, 3, 6, 2, 6, 4, 2, 3, 4, 3, 3]

Output : The odd occurring element is 4.

public class fundNum{

    public static int powOfTwo(int[] nums){

        int xor = 0;

        for (int i: nums) {

            xor = xor ^ i;

        }

        return xor;

    }

    public static void main(String[] args) {

        int []nums = {4,3,6,2,6,4,2,3,4,3,3};

        powOfTwo(nums);

        System.out.println("The odd occuring element is : " +powOfTwo(nums));

    }

}