1) Given a number print its binary representation.

Ans:

```
 import java.util.Stack;

public class Binary {

    public static void main(String[] args) {
        int num = 10;

        Stack<Integer> st = new Stack<>();

        if (num == 0) {
            System.out.println("0");
        }
        num = Math.abs(num);
        while (num != 0) {
            int rem = num % 2;
            st.push(rem);
            num = num / 2;
        }

        while (!st.isEmpty()) {
            System.out.print(st.pop());
        }
    }

}
```

2) Given a number ‘n’, predict it is a power of two or not.

Ans:

```
public class PowerOf2 {

    public static boolean predict(int num) {
        if (num <= 0) {
            return false;
        }
        int val = 1;
        while (true) {
```

```
            if (val < num) {
                val = val * 2;
            } else if (val > num) {
                return false;
            } else {
                return true;
            }
        }
    }

    public static void main(String[] args) {

        int num = 256;
        System.out.println(predict(num));

    }
}
```

3) Given a number. Using bit manipulation. Check whether it is odd or even.

Ans:

```
public class OddEvenChecker {
    public static void main(String[] args) {
        int number = 8;
        if ((number & 1) == 0) {
            System.out.println(number + " is even.");
        } else {
            System.out.println(number + " is odd.");
        }
    }
}
```

4) 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

Ans:

```
public class CountSetBits {
    public static void main(String[] args) {
        int number = 7; // Replace this with the number for which you want to count set bits

        int count = 0;

        while (number > 0) {
            // Use bitwise AND to check the least significant bit
            if ((number & 1) == 1) {
                count++;
            }

            // Right shift the number to check the next bit
            number >>= 1;
        }

        System.out.println("Number of set bits in " + number + " is " + count);
    }
}
```

5) 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.

Ans:

```
public class Duplicate {
    public static void main(String[] args) {
        int arr[] = { 4, 3, 4, 3, 3 };

        int ans = arr[0];
        for (int i = 1; i < arr.length; i++) {
            ans = ans ^ arr[i];
        }
```

```
        System.out.println(ans);
    }
}
```