ASSIGNMENT- (BIT-MANIPULATION)

Q1. given a number, print its binary representation.

Soln.

```java
import java.util.Scanner;

public class bitManipulation_Assg1 {

    static void binaryRepresentation(int n) {
        int result = 0;
        int base = 1;
        while (n > 0) {
            int rem = n % 2;
            result += rem * base;
            n = n / 2;
            base *= 10;
        }
        System.out.println(result);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number: ");
        int n = sc.nextInt();
        binaryRepresentation(n);
    }
}
```

Q2. given a number ‘n’, predict whether it is a power of two or not.

Soln.

```java
import java.util.Scanner;

public class bitManipulation_Assg2 {

    static int powerOfTwo(int n) {
        int result = 0;
        int base = 1;
```

```
        while (n > 0) {
            int rem = n % 2;
            result += rem * base;
            n = n / 2;
            base *= 10;
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number: ");
        int n = sc.nextInt();
        int result = powerOfTwo(n);
        if ((result & 1) == 0) {
            System.out.println(n + " is a power of two");
        } else {
            System.out.println(n + " is not a power of two");
        }
    }
}
```

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

Soln.

```
import java.util.Scanner;

public class bitManipulation_Assg3 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number: ");
        int n = sc.nextInt();
        if ((n & 1) == 1) {
            System.out.println(n + " is odd");
        } else {
            System.out.println(n + " is even");
        }
    }
}
```

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

Soln.

```
import java.io.*;
import java.util.*;
import java.util.Scanner;

public class bitManipulation_Assig4 {
    public static int countSetBits(int n) {
        int count = 0;
        while (n > 0) {
            count += n & 1;
            n >>= 1;
        }
        return count;
    }

    public static void main(String[] args) {
        int number;
        System.out.println("Enter the integer: ");

        // Create Scanner object
        Scanner s = new Scanner(System.in);

        // Read the next integer from the screen
        number = s.nextInt();

        int answer = countSetBits(number);
        System.out.println("The number of set bits in the given
number are " + answer);
    }
}
```

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.

Soln.

```
import java.io.*;
import java.util.*;
import java.util.Scanner;
```

```java
public class bitManipulation_Assg5 {
    public static int findOddOccuring(int[] arr) {
        int xor = 0;
        for (int i : arr) {
            xor = xor ^ i;
        }
        return xor;
    }

    public static void main(String[] args) {
        int n;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of elements you want to
store: ");
        // reading the number of elements from the that we want to
enter
        n = sc.nextInt();
        // creates an array in the memory of length 10
        int[] array = new int[10];
        System.out.println("Enter the elements of the array: ");
        for (int i = 0; i < n; i++) {
            // reading array elements from the user
            array[i] = sc.nextInt();
        }
        System.out.println("The odd occurring element is " +
findOddOccuring(array));
    }
}
```