DSA(number system)
Assignment question

Problem 1: given a number, print its binary representation. [easy]
Input 1: number = 5
Output 1: 101
Input 2: number = 10
Output 2: 1010

Ans:
```
public class Main {
   public static void main(String[] args) {
      int number = 5;
      System.out.println("Binary representation of " + number + ": " +
getBinaryRepresentation(number));

      number = 10;
      System.out.println("Binary representation of " + number + ": " +
getBinaryRepresentation(number));
   }

   public static String getBinaryRepresentation(int number) {
      return Integer.toBinaryString(number);
   }
}
```

Output:

```
Binary representation of 5: 101
Binary representation of 10: 1010
```

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

Ans:
```
public class Main {
    public static void main(String[] args) {
        int n = 15;
        System.out.println(isPowerOfTwo(n)); // False

        n = 32;
        System.out.println(isPowerOfTwo(n)); // True
    }

    public static boolean isPowerOfTwo(int n) {
        if (n <= 0) {
            return false;
        }
        return (n & (n - 1)) == 0;
    }
}
```

Q3. Problem 1: Given a number. Using bit manipulation, check whether it is odd or even.
Input 8, Even
3, False

Ans:

```
public class Main {
    public static void main(String[] args) {
        int num = 8;
        System.out.println(isEven(num)); // true

        num = 3;
        System.out.println(isEven(num)); // false
    }

    public static boolean isEven(int num) {
        return (num & 1) == 0;
    }
}
```

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.
Ans

```
public class Main {
    public static void main(String[] args) {
        int num = 15; // Binary: 1111
        System.out.println(countSetBits(num)); // Output: 4

        num = 10; // Binary: 1010
        System.out.println(countSetBits(num)); // Output: 2
    }

    public static int countSetBits(int num) {
        int count = 0;
        while (num != 0) {
            num = num & (num - 1); // Clear the least significant set bit
            count++;
        }
        return count;
    }
}
```

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.
Ans:

```
public class Main {
    public static void main(String[] args) {
        int[] arr = {4, 3, 6, 2, 6, 4, 2, 3, 4, 3, 3};
        System.out.println("The odd occurring element is " +
findOddElement(arr));
    }

    public static int findOddElement(int[] arr) {
        int result = 0;
        for (int num : arr) {
            result ^= num; // XOR operation
        }
        return result;
    }
}
```