Q1. Given an array. FInd the number X in the array. If the element is present, return the index of the element, else print “Element not found in array”. Input the size of array, array from user and the element X from user. Use Linear Search to find the element.

CODE:-

```
import java.util.*;
class FindingX{
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the size of the array : ");
        int n=sc.nextInt();
        System.out.println();
        int arr[]=new int[n];
        System.out.println("Enter the array elements : ");
        for(int i=0;i<n;i++){
            arr[i]=sc.nextInt();
        }
        System.out.print("Enter the element to search : ");
        int X=sc.nextInt();
        int f=0;
        for(int i=0;i<n;i++){
            if(arr[i]==X){
                f=1;
                    System.out.println("The element "+X+" is present at
index "+i);
                break;
            }

        }
        if(f==0){
            System.out.println("Element not found in the array.");
        }
    }
}
```

OUTPUT:-

Enter the size of the array : 5

Enter the array elements :

```
6
8
9
2
1
Enter the element to search : 9
The element 9 is present at index 2
```

Q2. Given an array and an integer “target”, return the last occurrence of “target” in the array. If the target is not present return -1.

CODE:-

```java
import java.util.*;
public class LastOcurrence {
    public static int binary_search(int arr[],int X){
        int low=0;
        int high=arr.length-1;
        int re=-1;
        while(low<=high){
            int mid=low + (high-low)/2;
            if(arr[mid]==X){
                re= mid;
                high=mid-1;
            }
            else if(arr[mid]>X){
                high=mid-1;
            }
            else{
                low=mid+1;
            }
        }
        return re;
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the size of the array : ");
        int n=sc.nextInt();
        System.out.println();
        int arr[]=new int[n];
        System.out.println("Enter the array elements : ");
        for(int i=0;i<n;i++){
            arr[i]=sc.nextInt();
        }
        System.out.print("Enter the element to search : ");
        int X=sc.nextInt();
        System.out.println();
        int result=binary_search(arr,X);
        if(result==-1){
            System.out.println(result);
        }
        else{
             System.out.println("The element "+X+" is present at index :
"+result);
        }
    }
}
```

OUTPUT:-

Enter the size of the array : 6

Enter the array elements :
2 7 7 8 8 10
Enter the element to search : 7

The element 7 is present at index : 1

Q3. Given a sorted binary array, efficiently count the total number of 1’s in it.

```java
import java.util.*;
public class CountOnes {
    public static int binary_search(int arr[],int X){
        int low=0;
        int high=arr.length-1;
        int re=-1;
        while(low<=high){
            int mid=low+(high-low)/2;
            if(arr[mid]==X){
                re= mid;
                high=mid-1;
            }
            else if(arr[mid]>X){
                high=mid-1;
            }
            else{
                low=mid+1;
            }
        }
        return re;
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the sixe of the array : ");
        int n=sc.nextInt();
        System.out.println();
        int arr[]=new int[n];
        System.out.println("Enter the array elements : ");
        for(int i=0;i<n;i++){
            arr[i]=sc.nextInt();
        }
        System.out.print("Enter the element to to count : ");
        int X=sc.nextInt();
        System.out.println();
        int result=binary_search(arr,X);
        int sum=0;
        for(int z=result;z<n;z++){
            if(arr[z]==X){
                sum++;
            }
        }
        if(result==-1){
            System.out.println(result);
        }
        else{
               System.out.println("The element "+X+" is present "+sum+"
times in the array");
        }
    }
}
```

OUTPUT:-

```
Enter the sixe of the array : 10

Enter the array elements :
0 0 0 0 1 1 1 1 1 1
Enter the element to to count : 1

The element 1 is present 6 times in the array
```

Q4. Given a sorted integer array containing duplicates, count occurrences of a given number. If the element is not found in the array, report that as well.

CODE:-

```java
import java.util.*;
public class CountNums {
    public static int binary_search(int arr[],int X){
        int low=0;
        int high=arr.length-1;
        int re=-1;
        while(low<=high){
            int mid=low+(high-low)/2;
            if(arr[mid]==X){
                re= mid;
                high=mid-1;
            }
            else if(arr[mid]>X){
                high=mid-1;
            }
            else{
                low=mid+1;
            }
        }
        return re;
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the sixe of the array : ");
        int n=sc.nextInt();
        System.out.println();
        int arr[]=new int[n];
        System.out.println("Enter the array elements : ");
        for(int i=0;i<n;i++){
            arr[i]=sc.nextInt();
        }
        System.out.print("Enter the element to to count : ");
        int X=sc.nextInt();
        System.out.println();
        int result=binary_search(arr,X);
        int sum=0;
        for(int z=result;z<n;z++){
            if(arr[z]==X){
                sum++;
            }
        }
        if(result==-1){
            System.out.println(result);
        }
        else{
               System.out.println("The element "+X+" is present "+sum+"
times in the array");
        }
    }
}
```

OUTPUT:-

```
Enter the sixe of the array : 10

Enter the array elements :
2 5 5 5 6 6 8 9 9 9
Enter the element to to count : 5

The element 5 is present 3 times in the array
```

Q5: Given a positive integer num, return true if num is a perfect square or false otherwise. A perfect square is an integer that is the square of an integer. In other words, it is the product of some integer with itself.

CODE:-

```


import java.util.*;
public class SquareRoot {
```

```java
    public static int square_root(int n){
        int low=0;
        int high=n;
        while(low<=high){
            int mid=low+(high-low)/2;
            int val=mid*mid;
            if(val==n){
                return mid;
            }
            else if(val<n){
                low=mid+1;
            }
            else{
                high=mid-1;
            }
        }
        return -1;
    }
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
         System.out.print("Enter a positive integer to determine whether
it is a perfect square or not : ");
        int n=sc.nextInt();
        System.out.println();
        int re=square_root(n);
        if(re==-1){
            System.out.println(false);
        }
        else{
            System.out.println(true);
        }
    }
}
```

OUTPUT:-

```
Enter a positive integer to determine whether it is a perfect square or not : 5
false
Enter a positive integer to determine whether it is a perfect square or not : 25
true
```