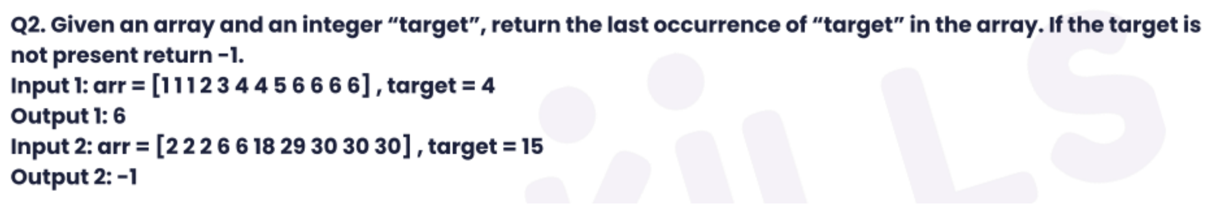
Q1. Given an array. FInd the number X in the array. If the element is present, return the index of the element, else print “Element not found in array”. Input the size of array, array from user and the element X from user. Use Linear Search to find the element.

Ans:

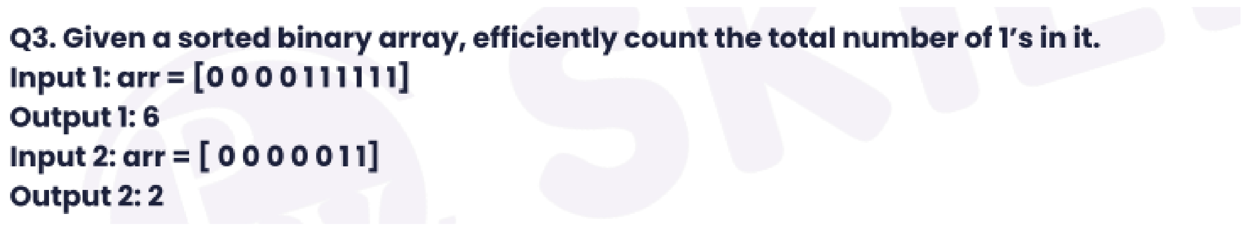
```
import java.util.Scanner;

public class LinearSearch {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the size of the array: ");
        int size = scanner.nextInt();

        int[] arr = new int[size];

        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < size; i++) {
            arr[i] = scanner.nextInt();
        }

        System.out.print("Enter the element to search (X): ");
        int X = scanner.nextInt();

        int index = linearSearch(arr, X);

        if (index != -1) {
            System.out.println("Element found at index: " + index);
        } else {
            System.out.println("Element not found in array.");
        }

        scanner.close();
    }

    // Linear search function
    public static int linearSearch(int[] arr, int X) {
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == X) {
                return i; // Element found, return its index
            }
        }
        return -1; // Element not found in the array
    }
}
```



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.
Input 1: arr = [1 1 1 2 3 4 4 5 6 6 6 6] , target = 4
Output 1: 6
Input 2: arr = [2 2 2 6 6 18 29 30 30 30] , target = 15
Output 2: -1

Ans:

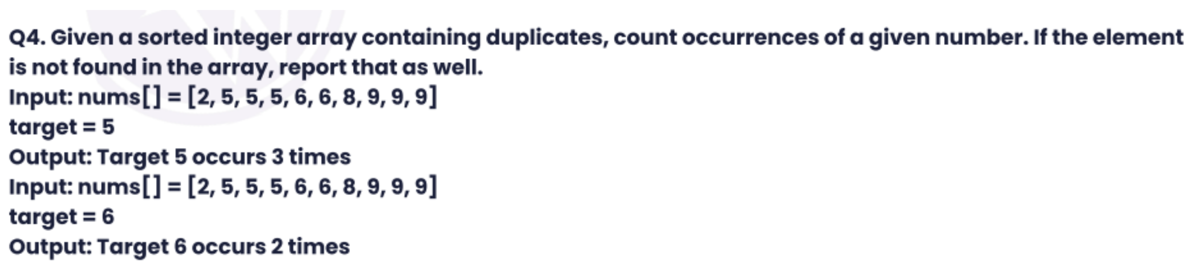
```
public class LastOccurrence {
    public static void main(String[] args) {

        int[] arr = {1, 2, 3, 4, 2, 5, 2}; // Replace this with your array
        int target = 2; // Replace this with your target integer

        int lastIndex = findLastOccurrence(arr, target);

        if (lastIndex != -1) {
            System.out.println("Last occurrence of " + target + " is at index " + lastIndex);
        } else {
            System.out.println(target + " is not present in the array.");
        }
    }

    public static int findLastOccurrence(int[] arr, int target) {
        for (int i = arr.length - 1; i >= 0; i--) {
            if (arr[i] == target) {
                return i;
            }
        }
        return -1; // Target not found in the array
    }
}
```



Q3. Given a sorted binary array, efficiently count the total number of 1’s in it.
Input 1: arr = [0 0 0 0 1 1 1 1 1 1]
Output 1: 6
Input 2: arr = [0 0 0 0 0 1 1]
Output 2: 2

Ans:

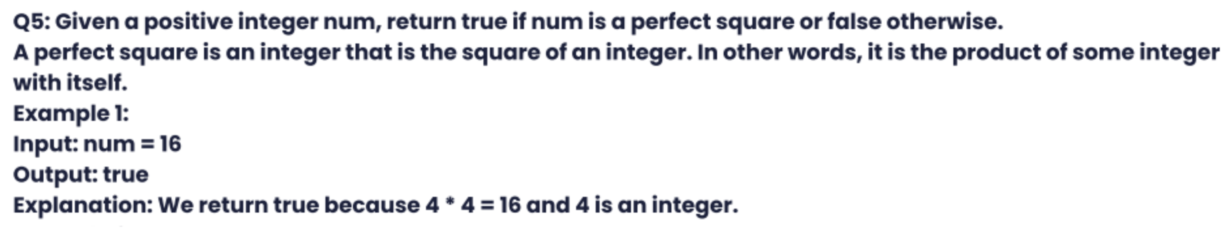
```
public class Binary {

    public static int oneCount(int[] arr) {

        int s = 0;
        int e = arr.length - 1;
        int mid = (s + e) / 2;
        while (s <= e) {
            mid = (s + e) / 2;
            if (arr[mid] == 0) {
                s = mid + 1;
            } else {
                e = mid - 1;
            }
        }

        if (s >= arr.length) {
            return -1;
        }
        return s;
    }

    public static void main(String[] args) {

        int arr[] = { 0, 0, 0, 0, 1, 1,1,1 };
        System.out.println(oneCount(arr));
    }

}
```



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.
Input: nums[] = [2, 5, 5, 5, 6, 6, 8, 9, 9, 9]
target = 5
Output: Target 5 occurs 3 times
Input: nums[] = [2, 5, 5, 5, 6, 6, 8, 9, 9, 9]
target = 6
Output: Target 6 occurs 2 times

Ans:

```
public class Main {
    public static void main(String[] args) {
        int[] sortedArray = {1, 2, 2, 2, 3, 4, 4, 5, 6};
        int target = 2;

        int count = countOccurrences(sortedArray, target);

        if (count != -1) {
            System.out.println("The number " + target + " appears " + count + " times in the array.");
        } else {
            System.out.println("The number " + target + " is not found in the array.");
        }
    }

    public static int countOccurrences(int[] sortedArray, int target) {
        int firstIndex = findFirstIndex(sortedArray, target);
        if (firstIndex == -1) {
            return -1; // Element not found in the array
        }

        int lastIndex = findLastIndex(sortedArray, target);
        return lastIndex - firstIndex + 1;
    }

    public static int findFirstIndex(int[] sortedArray, int target) {
        int low = 0;
        int high = sortedArray.length - 1;
        int result = -1;

        while (low <= high) {
            int mid = low + (high - low) / 2;

            if (sortedArray[mid] == target) {
                result = mid;
                high = mid - 1;
            } else if (sortedArray[mid] < target) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }

        return result;
    }

    public static int findLastIndex(int[] sortedArray, int target) {
        int low = 0;
        int high = sortedArray.length - 1;
        int result = -1;

        while (low <= high) {
            int mid = low + (high - low) / 2;

            if (sortedArray[mid] == target) {
                result = mid;
                low = mid + 1;
            } else if (sortedArray[mid] < target) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }

        return result;
    }
}
```



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.
Example 1:
Input: num = 16
Output: true
Explanation: We return true because 4 * 4 = 16 and 4 is an integer.

Ans:

```
public class checkSquare {

    public static boolean checkSq(int num) {

        if (num < 1) {
            return false;
        }

        if (num == 1) {
            return true;
        }

        int i = 1;
        int j = num - 1;
        int mid = (i + j) / 2;
        while (i < j) {
            mid = (i + j) / 2;
            if (mid * mid == num) {
                return true;
            } else if (mid * mid < num) {
                i = mid + 1;
            } else {
                j = mid - 1;
            }
        }

        return false;
    }

    public static void main(String[] args) {

        int num = 9;
        System.out.println(checkSq(num));
    }

}
```