Q1: Linear Search to Find Element in Array

```
import java.util.Scanner;

public class LinearSearch {

    public static int linearSearch(int[] array, int x) {
        for (int i = 0; i < array.length; i++) {
            if (array[i] == x) {
                return i; // Element found, return its index
            }
        }
        return -1; // Element not found
    }

    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[] array = new int[size];

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

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

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

        scanner.close();
    }
}
```

Q2: Last Occurrence of a Target in Array

```
public class LastOccurrence {

    public static int findLastOccurrence(int[] arr, int target) {
        for (int i = arr.length - 1; i >= 0; i--) {
            if (arr[i] == target) {
                return i; // Return the index of the last occurrence
            }
        }
        return -1; // Target not found
    }

    public static void main(String[] args) {
        int[] arr1 = {1, 1, 1, 2, 3, 4, 4, 5, 6, 6, 6, 6};
        int[] arr2 = {2, 2, 2, 6, 6, 18, 29, 30, 30, 30};

        System.out.println("Last occurrence of 4: " + findLastOccurrence(arr1, 4)); // Output: 6
        System.out.println("Last occurrence of 15: " + findLastOccurrence(arr2, 15)); // Output: -1
    }
}
```

Q3: Count Number of 1’s in Sorted Binary Array

```
public class CountOnes {

    public static int countOnes(int[] arr) {
        int low = 0, high = arr.length - 1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (arr[mid] == 1) {
                // Check for the first occurrence of 1
                if (mid == 0 || arr[mid - 1] == 0) {
                    return arr.length - mid;
                }
                high = mid - 1;
            } else {
                low = mid + 1;
            }
        }
        return 0; // No 1's found
    }

    public static void main(String[] args) {
        int[] arr1 = {0, 0, 0, 0, 1, 1, 1, 1, 1, 1};
        int[] arr2 = {0, 0, 0, 0, 0, 1, 1};

        System.out.println("Number of 1's in arr1: " + countOnes(arr1)); // Output: 6
        System.out.println("Number of 1's in arr2: " + countOnes(arr2)); // Output: 2
    }
}
```

Q4: Count Occurrences of a Given Number in Sorted Array

```
public class CountOccurrences {

    public static int countOccurrences(int[] nums, int target) {
        int first = findFirstOccurrence(nums, target);
        int last = findLastOccurrence(nums, target);

        if (first == -1) {
            System.out.println("Target not found");
            return 0;
        }

        return last - first + 1;
    }

    private static int findFirstOccurrence(int[] nums, int target) {
        int low = 0, high = nums.length - 1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (nums[mid] < target) {
                low = mid + 1;
            } else if (nums[mid] > target) {
                high = mid - 1;
            } else {
                if (mid == 0 || nums[mid - 1] != target) {
                    return mid;
                }
                high = mid - 1;
            }
        }
        return -1;
    }

    private static int findLastOccurrence(int[] nums, int target) {
        int low = 0, high = nums.length - 1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (nums[mid] < target) {
                low = mid + 1;
            } else if (nums[mid] > target) {
                high = mid - 1;
            } else {
                if (mid == nums.length - 1 || nums[mid + 1] != target) {
                    return mid;
                }
                low = mid + 1;
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        int[] nums1 = {2, 5, 5, 5, 6, 6, 8, 9, 9, 9};
        int[] nums2 = {2, 5, 5, 5, 6, 6, 8, 9, 9, 9};

        System.out.println("Target 5 occurs " + countOccurrences(nums1, 5) + " times"); // Output:
3
        System.out.println("Target 6 occurs " + countOccurrences(nums2, 6) + " times"); // Output:
2
    }
}
```

Q5: Check if a Number is a Perfect Square

```
public class PerfectSquare {

    public static boolean isPerfectSquare(int num) {
        if (num < 0) return false;
        int sqrt = (int) Math.sqrt(num);
        return sqrt  sqrt == num;
    }

    public static void main(String[] args) {
        System.out.println("Is 16 a perfect square? " + isPerfectSquare(16)); // Output: true
        System.out.println("Is 14 a perfect square? " + isPerfectSquare(14)); // Output: false
    }
}
```