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.

import java.util.Scanner;

public class linearSearchAssign {

    public static int linarSearch(int []arr, int target){

        for(int i = 0;i<arr.length;i++){

            if(arr[i] == target){

                return i;

            }

        }

        return -1;

    }

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the size of an array :");

        int size = sc.nextInt();

        int arr[] = new int[size];

        System.out.println("Enter the values of an array : ");

        for(int i = 0;i<arr.length;i++){

            arr[i] =sc.nextInt();

        }

        System.out.println("Enter the key element you want to find :");

        int key = sc.nextInt();

        int result = linarSearch(arr, key);

        if(result == -1){

            System.out.println("Element not found in array");

        }

        else{

            System.out.println("Element found in index : "+result);

        }

    }

}

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

import java.util.Scanner;

public class linBinAssign3 {

    public static int binarySearch(int []arr, int target){

        int low = 0;

        int high = arr.length-1;

        int result = -1;

        while(low<=high){

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

            if(arr[mid] == target){

                result = mid;

                low = mid+1;

            }

            else if(arr[mid]> target){

                high = mid-1;

            }

            else{

                low = mid+1;

            }

        }

        return result;

    }

    public static void main(String[] args) {

        Scanner sc  = new Scanner(System.in);

        System.out.println("enter the size of an array : ");

        int size = sc.nextInt();

        int arr[] = new int[size];

        System.out.println("Enter the values of an array :");

        for(int i = 0;i<arr.length;i++){

             arr[i] = sc.nextInt();

        }

        System.out.println("Enter the value of you want to find : ");

        int x = sc.nextInt();

        int result = binarySearch(arr, x);

        if(result == -1){

            System.out.println("The element not found");

        }

        else{

            System.out.println("The element found at index : "+result);

        }

}

}

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

import java.util.Scanner;

public class linBinAssign3 {

    public static int binarySearch(int [] arr, int target){

        int high = arr.length-1;

        int low = 0;

        int result = -1;

        while(low<=high){

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

            if(arr[mid] == target){

                result = mid;

                high = mid-1;

            }

            else if(arr[mid]>target){

                high = mid-1;

            }

            else{

                low = mid+1;

            }

            result = arr.length - result;

        }

        return result;

    }

    public static void main(String[] args) {

        Scanner sc  = new Scanner(System.in);

        System.out.println("enter the size of an array : ");

        int size = sc.nextInt();

        int arr[] = new int[size];

        System.out.println("Enter the values of an array :");

        for(int i = 0;i<arr.length;i++){

             arr[i] = sc.nextInt();

        }

        System.out.println("Enter the value of you want to find : ");

        int x = sc.nextInt();

        int result = binarySearch(arr, x);

        if(result == -1){

            System.out.println("The element not found");

        }

        else{

            System.out.println("The number of elements are : "+result);

        }

    }

}

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

import java.util.Scanner;

public class linBinAssign4 {

        public static int lowIndex(int [] arr, int target){

            int high = arr.length-1;

            int low = 0;

            int result = -1;

            while(low<=high){

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

                if(arr[mid] == target){

                    result = mid;

                    high = mid-1;

                }

                else if(arr[mid]>target){

                    high = mid-1;

                }

                else{

                    low = mid+1;

                }

            }

            return result;

        }

        public static int highestIndex(int [] arr, int target){

            int high = arr.length-1;

            int low = 0;

            int result = -1;

            while(low<=high){

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

                if(arr[mid] == target){

                    result = mid;

                    low = mid+1;

                }

                else if(arr[mid]>target){

                    high = mid-1;

                }

                else{

                    low = mid+1;

                }

            }

            return result;

        }

        public static void main(String[] args) {

            Scanner sc  = new Scanner(System.in);

            System.out.println("enter the size of an array : ");

            int size = sc.nextInt();

            int arr[] = new int[size];

            System.out.println("Enter the values of an array :");

            for(int i = 0;i<arr.length;i++){

                 arr[i] = sc.nextInt();

            }

            System.out.println("Enter the value of you want to find : ");

            int x = sc.nextInt();

            int highIndex = highestIndex(arr, x);

            int lowIndex = lowIndex(arr, x);

            System.out.println("Target "+x+" occurs "+((highIndex-lowIndex)+1)+" times");

        }

    }

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.

Example 2:

Input: num = 14

Output: false

Explanation: We return false because 3.742 \* 3.742 = 14 and 3.742 is not an integer.

import java.util.Scanner;

public class linBinAssign5 {

        public static int squareRoot(int num){

            int start = 0;

            int end = num/2;

            while(start<=end){

                int mid = start +(end-start)/2;

                if(mid\*mid ==  num){

                    return mid;

                }

                else if(mid\*mid>num){

                    end = mid-1;

                }

                else{

                    start = mid+1;

                }

            }

            return -1;

        }

        public static void main(String[] args) {

            Scanner sc  = new Scanner(System.in);

            System.out.println("enter the value you want perfect square of : ");

            int num = sc.nextInt();

            int squareRoot = squareRoot(num);

            if(squareRoot == -1)

            System.out.println("False");

            else

            System.out.println("True");

        }

    }