Searching in an array- Linear

& Binary Search

Assignment question

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 Main {
    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 " + size + " elements:");
        for (int i = 0; i < size; i++) {
            array[i] = scanner.nextInt();
        }

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

        int result = linearSearch(array, x);

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

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

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:

```
import java.util.Scanner;

public class Main {
    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 " + size + " elements:");
        for (int i = 0; i < size; i++) {
            array[i] = scanner.nextInt();
        }

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

        int result = lastOccurrence(array, target);

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

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

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:
```
import java.util.Scanner;

public class Main {
    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 " + size + " elements:");
        for (int i = 0; i < size; i++) {
            array[i] = scanner.nextInt();
        }

        int count = countOnes(array);

        System.out.println("Total number of 1's: " + count);
    }

    public static int countOnes(int[] array) {
        int low = 0;
        int high = array.length - 1;

        // Find the first occurrence of 1 using Binary Search
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (array[mid] == 0) {
                low = mid + 1;
            } else if (mid == 0 || array[mid - 1] == 0) {
                return array.length - mid; // Count 1's from mid to end
            } else {
                high = mid - 1;
            }
        }
        return 0; // No 1's found
    }
}
```

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:

```
import java.util.Scanner;

public class Main {
    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[] nums = new int[size];
        System.out.println("Enter " + size + " elements:");
        for (int i = 0; i < size; i++) {
            nums[i] = scanner.nextInt();
        }

        System.out.print("Enter the target element: ");
        int target = scanner.nextInt();
        int count = countOccurrences(nums, target);

        if (count == 0) {
            System.out.println("Target " + target + " not found in array");
        } else {
            System.out.println("Target " + target + " occurs " + count + "
times");
        }
    }

    public static int countOccurrences(int[] nums, int target) {
        int first = findFirstOccurrence(nums, target);
        if (first == -1) {
            return 0;
        }

        int last = findLastOccurrence(nums, target);
        return last - first + 1;
    }

    public static int findFirstOccurrence(int[] nums, int target) {
        int low = 0;
        int 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;
            } else {
                high = mid - 1;
            }
        }
        return -1;
    }

    public static int findLastOccurrence(int[] nums, int target) {
        int low = 0;
        int 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;
            } else {
                low = mid + 1;
            }
        }

        return -1;
    }
}
```

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.

Ans:

```
public class Main {
    public static void main(String[] args) {
        System.out.println(isPerfectSquare(16)); // true
        System.out.println(isPerfectSquare(14)); // false
    }

    public static boolean isPerfectSquare(int num) {
        int sqrt = (int) Math.sqrt(num);
        return sqrt * sqrt == num;
    }
}
```