NAME: ANEEQ SHAMS ROLL NO: BSE-230 SECTION: E

DATA STRUCTURES AND ALGORITHMS

LAB NO # 06

OBJECTIVE: To find an element in linear array using Linear Search and Binary Search.

LAB TASKS :

1. **Declare an array of size 10 to store account balances. Initialize with values 0 to 1000000. Check all array if any value is less than 10000. Show message:**
   Account No. Low Balance
   Account No. Low Balance

INPUT :

```
import java.util.Random;

public class BankAccountBalanceCheck {
    public static void main(String[] args) {
        int n = 10; // Number of accounts
        int[] arrHussain = new int[n]; // Array to store account balances

        Random random = new Random();

        // Initialize array with random account balances between 0 and 1,000,000
        for (int i = 0; i < n; i++) {
            arrHussain[i] = random.nextInt(1000001); // Random balance between 0 and 1,000,000
        }

        // Display all account balances
        System.out.println("Account Balances (arrHussain):");
        for (int i = 0; i < n; i++) {
            System.out.println("Account No. " + (1000 + i) + " Balance: " + arrHussain[i]);
        }

        // Check for accounts with a balance less than 10,000 and display a message
        System.out.println("\nChecking for Low Balances...");
        for (int i = 0; i < n; i++) {
            if (arrHussain[i] < 10000) {
                System.out.println("Account No. " + (1000 + i) + " has Low Balance: " + arrHussain[i]);
            }
        }
    }
}
```

Output ;

```
run:
Account Balances (arrHussain):
Account No. 1000 Balance: 731311
Account No. 1001 Balance: 699281
Account No. 1002 Balance: 725984
Account No. 1003 Balance: 484338
Account No. 1004 Balance: 876440
Account No. 1005 Balance: 264192
Account No. 1006 Balance: 121341
Account No. 1007 Balance: 406178
Account No. 1008 Balance: 91030
Account No. 1009 Balance: 199769

Checking for Low Balances...
```

2. **Write a program to search in array using Array built-in class.**

Input :

```
import java.util.Arrays;
import java.util.Scanner;

public class SearchInArray {
    public static void main(String[] args) {
        // Initialize an array named arrHussain
        int[] arrHussain = {100,85,200,185,300,285,400,385};

        // Display the array
        System.out.println("Array (arrHussain): " + Arrays.toString(arrHussain));

        // Prompt the user for the value to search
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number to search in arrHussain: ");
        int searchValue = scanner.nextInt();

        // Sort the array (required for binary search)
        Arrays.sort(arrHussain);
        System.out.println("Sorted Array (arrHussain): " + Arrays.toString(arrHussain));

        // Search for the value using binary search
        int index = Arrays.binarySearch(arrHussain, searchValue);

        // Check if the value was found
        if (index >= 0) {
            System.out.println("Number " + searchValue + " found at position " + (index + 1) + " in the sorted array.");
        } else {
            System.out.println("Number " + searchValue + " not found in arrHussain.");
        }

        scanner.close();
    }
}
```

Output :

```
run:
Array (arrHussain): [100, 85, 200, 185, 300, 285, 400, 385]
Enter the number to search in arrHussain: 285
Sorted Array (arrHussain): [85, 100, 185, 200, 285, 300, 385, 400]
Number 285 found at position 5 in the sorted array.
BUILD SUCCESSFUL (total time: 6 seconds)
```

3. **Given an unsorted array arr of integers, find the smallest positive integer that is** missing **from the array. You need to implement this using** binary search**. The array can contain both negative numbers and positive numbers, and you can assume that the array does not have duplicates.**

INPUT :

NAME: ANEEQ SHAMS ROLL NO: BSE-230 SECTION: E

```java
import java.util.Arrays;

public class SmallestMissingPositive {
    public static void main(String[] args) {
        // Unsorted array named arrHussain
        int[] arrHussain = {4, -2, 1, 6, 8, -1};

        // Display the original array
        System.out.println("Original Array (arrHussain): " + Arrays.toString(arrHussain));

        // Step 1: Filter the array to include only positive numbers
        arrHussain = Arrays.stream(arrHussain).filter(x -> x > 0).toArray();

        // Step 2: Sort the filtered array
        Arrays.sort(arrHussain);
        System.out.println("Filtered & Sorted Array (arrHussain): " + Arrays.toString(arrHussain));

        // Step 3: Find the smallest missing positive integer using binary search
        int smallestMissing = findSmallestMissing(arrHussain);

        // Display the result
        System.out.println("The smallest missing positive integer is: " + smallestMissing);
    }

    // Function to find the smallest missing positive integer
    public static int findSmallestMissing(int[] arrHussain) {
        int left = 0, right = arrHussain.length - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;

            // Check if the current element is in its correct position
            if (arrHussain[mid] == mid + 1) {
                left = mid + 1; // Move to the right
            } else {
                right = mid - 1; // Move to the left
            }
        }

        // The smallest missing positive integer is left + 1
        return left + 1;
    }
}
```

Output :

```
run:
Original Array (arrHussain): [4, -2, 1, 6, 8, -1]
Filtered & Sorted Array (arrHussain): [1, 4, 6, 8]
The smallest missing positive integer is: 2
BUILD SUCCESSFUL (total time: 0 seconds)
```

4. **You are given a sorted array arr[] and a target element target. Your task is to find the** first **occurrence** of the target in the array using binary search. If the target is not found, return -1. You are given a sorted array arr[] and a target element target. Your task is to find the **first occurrence of the target in the array using binary search. If the target is not found**, **return -1**

INPUT :

```java
import java.util.Scanner;

public class FirstOccurrenceBinarySearch {
    public static void main(String[] args) {
        // Sorted array named arrHussain
        int[] arrHussain = {2, 4, 4, 10, 10, 10, 18, 20, 20};

        // Display the array
        System.out.println("Array (arrHussain): ");
        for (int num : arrHussain) {
            System.out.print(num + " ");
        }
        System.out.println();

        // Take target input from the user
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the target to find its first occurrence in arrHussain: ");
        int target = scanner.nextInt();

        // Find the first occurrence using binary search
        int firstOccurrence = findFirstOccurrence(arrHussain, target);

        // Print the result
        if (firstOccurrence != -1) {
            System.out.println("First occurrence of " + target + " is at index: " + firstOccurrence);
        } else {
            System.out.println("Target " + target + " not found in arrHussain.");
        }

        scanner.close();
    }
    // Function to find the first occurrence of the target
    public static int findFirstOccurrence(int[] arrHussain, int target) {
        int left = 0, right = arrHussain.length - 1;
        int result = -1; // Default to -1 if the target is not found

        while (left <= right) {
            int mid = left + (right - left) / 2;

            if (arrHussain[mid] == target) {
                result = mid; // Record the index
                right = mid - 1; // Continue searching on the left for the first occurrence
            } else if (arrHussain[mid] > target) {
                right = mid - 1; // Narrow down to the left half
            } else {
                left = mid + 1; // Narrow down to the right half
            }
        }

        return result;
    }
}
```

OUTPUT :

```
run:
Array (arrHussain):
2 4 4 10 10 10 18 20 20
Enter the target to find its first occurrence in arrHussain: 10
First occurrence of 10 is at index: 3
BUILD SUCCESSFUL (total time: 12 seconds)
```

HOME TASKS :

1. Write a program initializing array of size 20 and search an element using binary search.

INPUT :

```
import java.util.Scanner;
import java.util.Arrays;

public class BinarySearchArray {
    public static void main(String[] args) {
        // Initialize an array of size 20 with unique random numbers
        int[] arrHussain = {3, 7, 15, 18, 23, 28, 34, 39, 42, 50, 54, 61, 66, 72, 77, 84, 88, 93, 97, 101};

        // Display the array
        System.out.println("Array (arrHussain): " + Arrays.toString(arrHussain));

        // Input the target element to search
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the element to search in arrHussain: ");
        int target = scanner.nextInt();

        // Perform binary search
        int index = binarySearch(arrHussain, target);

        // Output the result
        if (index != -1) {
            System.out.println("Element " + target + " found at index: " + index);
        } else {
            System.out.println("Element " + target + " not found in arrHussain.");
        }

        scanner.close();
    }

    // Binary Search Function
    public static int binarySearch(int[] arrHussain, int target) {
        int left = 0, right = arrHussain.length - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;

            // Check if target is at the middle
            if (arrHussain[mid] == target) {
                return mid;
            }

            // Decide the direction to search
            if (arrHussain[mid] < target) {
                left = mid + 1; // Target is in the right half
            } else {
                right = mid - 1; // Target is in the left half
            }
        }

        return -1; // Target not found
    }
}
```

Output :

```
run:
Array (arrHussain): [3, 7, 15, 18, 23, 28, 34, 39, 42, 50, 54, 61, 66, 72, 77, 84, 88, 93, 97, 101]
Enter the element to search in arrHussain: 54
Element 54 found at index: 10
BUILD SUCCESSFUL (total time: 11 seconds)
```

2. **Write a function called occurrences that, given an array of numbers A, prints all the distinct values in A each followed by its number of occurrences.**
   For example, if A = (28, 1, 0, 1, 0, 3, 4, 0, 0, 3), the function should output the following five lines (here separated by a semicolon) "28 1; 1 2; 0 4; 3 2; 4 1".

INPUT :

```
import java.util.HashMap;

public class OccurrencesCounter {

    public static void main(String[] args) {
        // Define the array
        int[] arrHussain = {28, 1, 0, 1, 0, 3, 4, 0, 0, 3};

        // Call the occurrences function
        printOccurrences(arrHussain);
    }

    public static void printOccurrences(int[] arrHussain) {
        // Use HashMap to store values and their counts
        HashMap<Integer, Integer> frequencyMap = new HashMap<>();

        // Count occurrences
        for (int num : arrHussain) {
            frequencyMap.put(num, frequencyMap.getOrDefault(num, 0) + 1);
        }

        // Print each distinct value and its occurrence count
        System.out.println("Occurrences of elements in arrHussain:");
        StringBuilder result = new StringBuilder();
        for (int key : frequencyMap.keySet()) {
            result.append(key).append(" ").append(frequencyMap.get(key)).append("; ");
        }

        // Remove the last semicolon and print the result
        System.out.println(result.substring(0, result.length() - 2));
    }
}
```

Output :

```
run:
Occurrences of elements in arrHussain:
0 4; 1 2; 3 2; 4 1; 28 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. **Assume a bank's system needs to identify accounts with critically low balances and alert the user. Test the function with various balance values to ensure it correctly identifies all accounts below the threshold.**

INPUT :

```
import java.util.Random;

public class LowBalanceAlert {

    public static void main(String[] args) {
        // Declare an array with account balances
        int[] arrHussain = new int[10];

        // Generate random balances between 0 and 100000 for testing
        Random random = new Random();
        for (int i = 0; i < arrHussain.length; i++) {
            arrHussain[i] = random.nextInt(100001); // Random balances between 0 and 100000
        }

        // Set a threshold for critically low balances
        int lowBalanceThreshold = 10000;

        // Check for accounts with balances below the threshold
        printLowBalanceAccounts(arrHussain, lowBalanceThreshold);
    }

    // Function to print accounts with low balances
    public static void printLowBalanceAccounts(int[] arrHussain, int threshold) {
        System.out.println("Accounts with critically low balances (below " + threshold + "):");
        boolean foundLowBalance = false;

        // Loop through the array and check balances
        for (int i = 0; i < arrHussain.length; i++) {
            if (arrHussain[i] < threshold) {
                System.out.println("Account No. " + (i + 1) + " Balance: " + arrHussain[i]);
                foundLowBalance = true;
            }
        }
       if (!foundLowBalance) {
           System.out.println("No accounts with critically low balances.");
       }
    }
}
```

Output :

```
run:
Accounts with critically low balances (below 10000):
No accounts with critically low balances.
BUILD SUCCESSFUL (total time: 0 seconds)
```