LAB 06

Searching in Linear Array

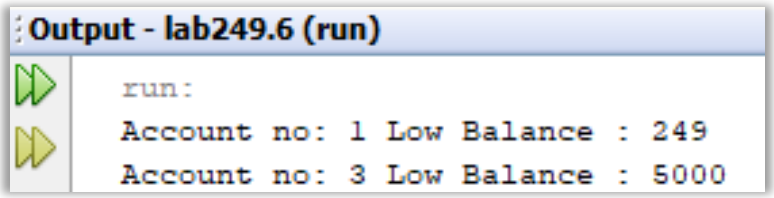
Lab Tasks

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

Source Code :

```
public class Lab2496 {
    public static void main(String[] args) {
     int[] amountbalance={249,10000,5000,10000};
     for(int i=0;i<amountbalance.length;i++){
         if(amountbalance[i]<10000){
System.out.println("Account no: "+(i+1)+"Low Balance"+amountbalance[i]);
         } }
     }
}
```

Output :

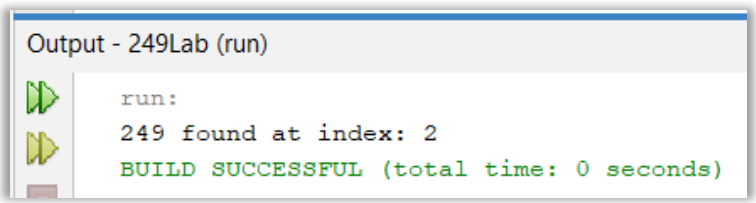


2. Write a program to search in array using Array built-in class.

Source Code :

```
public class lab249 {
    public static void main(String[] args) {
     int[] numbers = {2, 4, 249, 10, 0};
        int target = 249;
        int index = Arrays.binarySearch(numbers, target);
        if (index >= 0) {
            System.out.println(target + " found at index: " + index);
        } else {
            System.out.println(target + " not found in the array.");
        }
    }
}
```

Output :



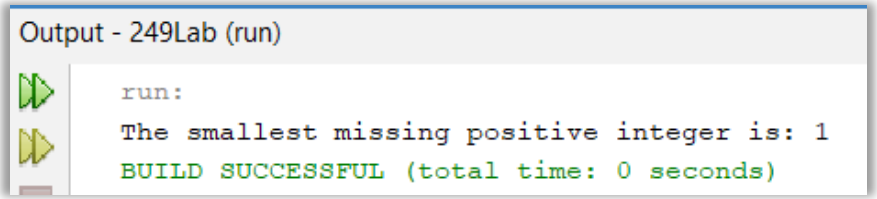
3. Given an unsorted array `arr` of integers, find the smallest positive integer that is **missing** from the array. You need to implement this using **binary search**. The array can contain both negative numbers and positive numbers, and you can assume that the array does not have duplicates.

Source Code :

```
public class lab249 {
    public static void main(String[] args) {
        int[] arr = {2, 4, -249, 249};
System.out.println("Smallest missing +ve int:" +findMissingPositive(arr));
    }
    public static int findMissingPositive(int[] arr) {
        int n = arr.length;
        int shift = segregate(arr, n);
        int[] positiveArr = Arrays.copyOfRange(arr, shift, n);
        return findMissing(positiveArr);
    }
    public static int segregate(int[] arr, int n) {
        int j = 0;
        for (int i = 0; i < n; i++) {
            if (arr[i] <= 0) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
                j++;
            }
        }
        return j;
    }

    public static int findMissing(int[] arr) {
        int n = arr.length;
        for (int i = 0; i < n; i++) {
            int val = Math.abs(arr[i]);
            if (val - 1 < n && arr[val - 1] > 0) {
                arr[val - 1] = -arr[val - 1];
            }
        }
        for (int i = 0; i < n; i++) {
            if (arr[i] > 0) return i + 1;
        }
        return n + 1;
    }
}
```

Output :



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`.

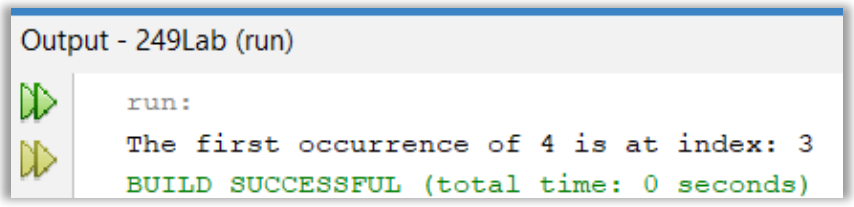
Source Code :

```
public class lab249 {
public static void main(String[] args) {
        int[] arr = {0, 2, 2, 4, 4, 9, 9};
        int target = 4;
        int index = findFirstOccurrence(arr, target);
        System.out.println("The first occurrence of " + target + " is at
index: " + index);
    }
    public static int findFirstOccurrence(int[] arr, int target) {
        int left = 0;
        int right = arr.length - 1;
        int result = -1;

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

            if (arr[mid] == target) {
                result = mid;
                right = mid - 1;
            } else if (arr[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return result;
    }
}
```

Output :



Home Tasks

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

Source Code :

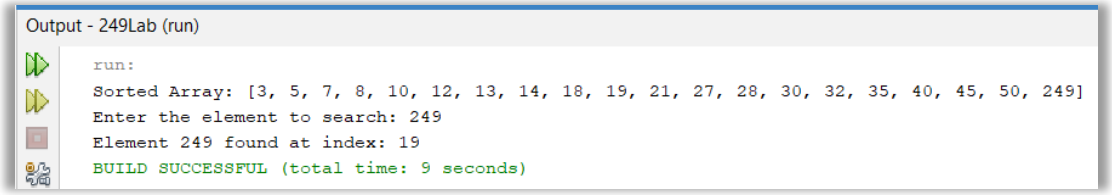
```
public class lab249 {
    public static void main(String[] args) {
        int[] arr = {249, 7, 13, 45, 32, 10, 28, 19, 50, 3, 8, 12, 30, 14, 18, 21,
40, 5, 27, 35};
        Arrays.sort(arr);
        System.out.println("Sorted Array: " + Arrays.toString(arr));

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

        int result = binarySearch(arr, target);
        System.out.println(result != -1 ? "Element " + target + " found at index:
" + result : "Element " + target + " not found.");
        scanner.close();
    }

    public static int binarySearch(int[] arr, int target) {
        int left = 0, right = arr.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (arr[mid] == target) return mid;
            if (arr[mid] < target) left = mid + 1;
            else right = mid - 1;
        }
        return -1;
    }
}
```

Output :



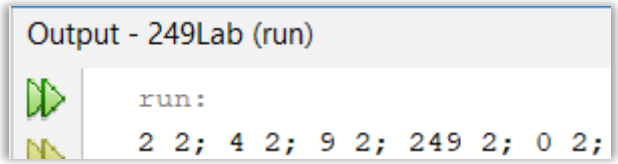
2. Write a function called occurrences that, given an array of numbers A, prints all the distinct values in A each followed by its number of occurrences. For eg, if A = (28, 1, 0, 1, 0, 3, 4, 0, 0, 3), function should output the following five lines (here separated by a semicolon) “28 1; 1 2; 0 4; 3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

Source Code :

```
public class lab249 {
    public static void main(String[] args) {
        int[] A = {2, 4, 9, 249, 0, 2, 9, 4, 249, 0};
        occurrences(A);
    } public static void occurrences(int[] A) {
        boolean[] counted = new boolean[A.length];
        for (int i = 0; i < A.length; i++) {
            if (!counted[i]) {
                int count = 1;
```

```
                for (int j = i + 1; j < A.length; j++) {
                    if (A[i] == A[j]) {
                        count++;
                        counted[j] = true;
                    }}System.out.print(A[i] + " " + count + "; ");
            }}}
```

Output :



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.

Source Code :

```
import java.util.ArrayList;
import java.util.List;
public class lab249 {
    public static void main(String[] args) {
        double[] balances = {500.0, 49.99, 250.5, 10.0, 75.0, 249.0,
1000.0, 249.9, 5.0};
        double criticalThreshold = 50.0;
        List<Integer> lowBalanceAccounts = identifyLowBalances(balances,
criticalThreshold);
                if (lowBalanceAccounts.isEmpty()) {
System.out.println("No accounts have critically low balances.");
        } else {
            System.out.println("Accounts with critically low balances:");
            for (int accountIndex : lowBalanceAccounts) {
                System.out.println("Account " + (accountIndex + 1) + " has
a balance of $" + balances[accountIndex]);
            }
        }
    }public static List<Integer> identifyLowBalances(double[] balances,
double threshold) {
        List<Integer> lowBalanceAccounts = new ArrayList<>();
        for (int i = 0; i < balances.length; i++) {
            if (balances[i] < threshold) {
                lowBalanceAccounts.add(i);
            }
        }return lowBalanceAccounts;
    }
}
```

Output :