Lab #06

Searching in a Linear Array

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

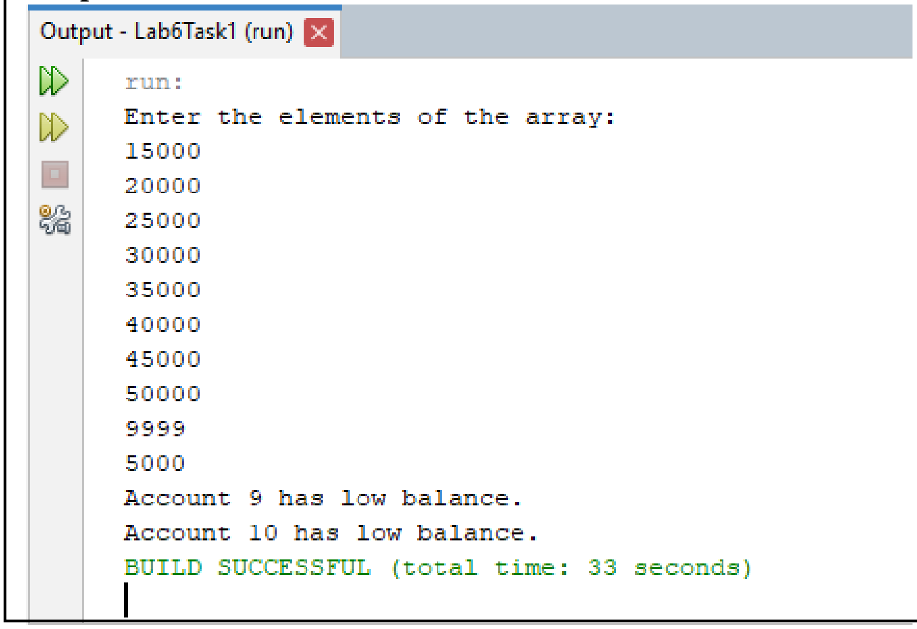
Lab Tasks:

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

Source Code:

```
import java.util.Scanner;
public class KhurramLab6Task1 {
    public static void main(String[] args) {
        int balance = 10000;
        int[] arr = new int[10];
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the elements of the array: ");
        for (int i = 0; i < arr.length; i++) arr[i] = sc.nextInt();
        for (int i = 0; i < arr.length; i++)
            if (arr[i] < balance)
                System.out.println("Account " + (i + 1) + " has low balance.");
        sc.close();}}
```

Output:



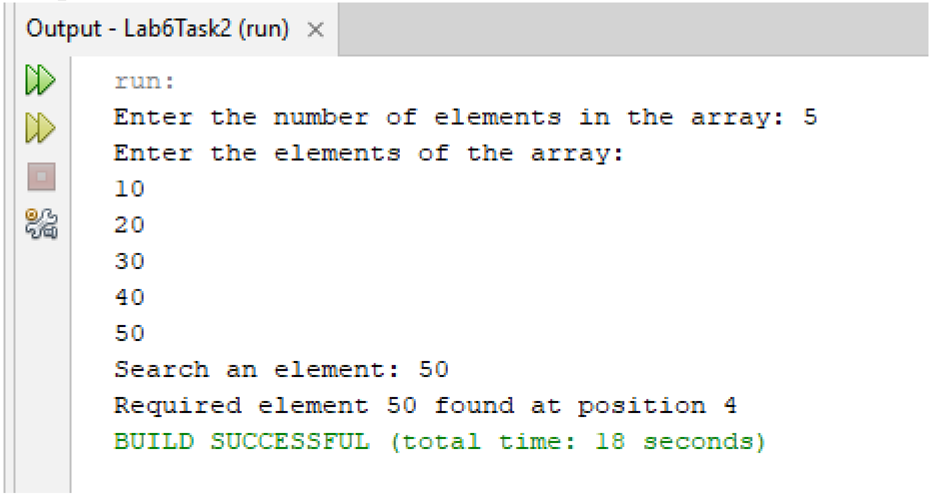
```
run:
Enter the elements of the array:
15000
20000
25000
30000
35000
40000
45000
50000
9999
5000
Account 9 has low balance.
Account 10 has low balance.
BUILD SUCCESSFUL (total time: 33 seconds)
```

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

Source Code:

```
import java.util.Scanner;
public class KhurramLab6Task2 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of elements in the array: ");
        int n = sc.nextInt();
        int[] a = new int[n];
        System.out.println("Enter the elements of the array: ");
        for (int i = 0; i < n; i++) a[i] = sc.nextInt();
        System.out.print("Search an element: ");
        int search = sc.nextInt();
        boolean found = false;
        for (int i = 0; i < n; i++) {
            if (a[i] == search) {
                System.out.println("Required element " + search + " found at position " + i);
                found = true;
                break;}}
        if (!found) System.out.println("Element " + search + " not found in the array.");
        sc.close();}}
```

Output:



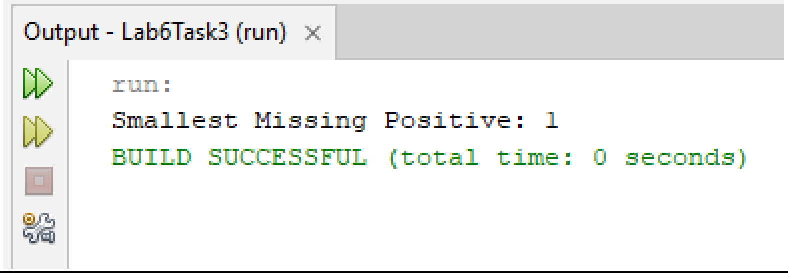
```
run:
Enter the number of elements in the array: 5
Enter the elements of the array:
10
20
30
40
50
Search an element: 50
Required element 50 found at position 4
BUILD SUCCESSFUL (total time: 18 seconds)
```

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.

Source Code:

```
import java.util.Arrays;
public class KhurramLab6Task3 {
   public static int findSmallestMissing(int[] arr) {
      Arrays.sort(arr);
      int left = 0, right = arr.length - 1;
      while (left <= right) {
         int mid = left + (right - left) / 2;
         if (arr[mid] <= 0 || arr[mid] != mid + 1) right = mid - 1;
         else left = mid + 1;}
      return left + 1;}
   public static void main(String[] args) {
      int[] arr = {3, 4, -1, 1};
      System.out.println("Smallest Missing Positive: " + findSmallestMissing(arr));}}
```

Output:



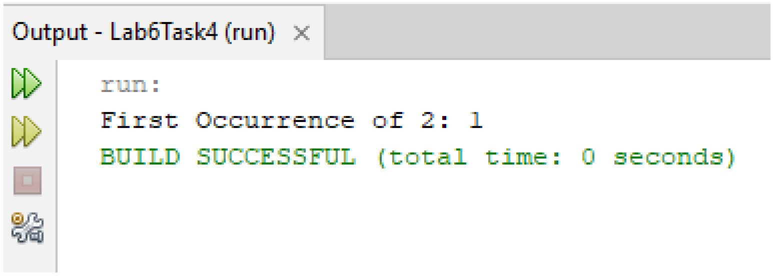
```
run:
Smallest Missing Positive: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

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.

Source Code:

```
package lab6task4;
public class KhurramLab6Task4 {
   public static int findFirstOccurrence(int[] arr, int target) {
      int left = 0, right = arr.length - 1, 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;}
   public static void main(String[] args) {
      int[] arr = {1, 2, 2, 3, 4, 5};
      int target = 2;
      System.out.println("First Occurrence of " + target + ": " + findFirstOccurrence(arr, target));}}
```

Output:



```
run:
First Occurrence of 2: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

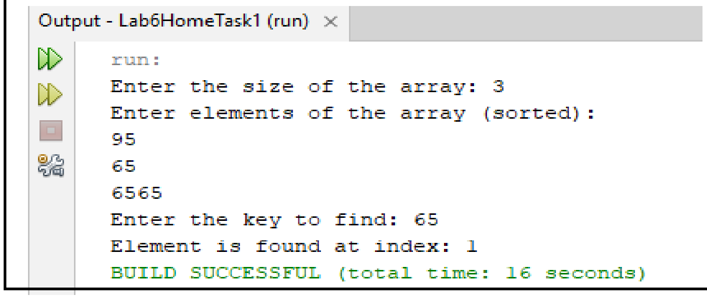
Home Tasks:

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

Source Code:

```
import java.util.*;
public class KhurramLab6HomeTask1 {
    public static void binarySearch(int[] arr, int key) {
        int first = 0, last = arr.length - 1;
        while (first <= last) {
            int mid = (first + last) / 2;
            if (arr[mid] < key) {
                first = mid + 1;
            } else if (arr[mid] == key) {
                System.out.println("Element is found at index: " + mid);
                return;
            } else {
                last = mid - 1;}}
        System.out.println("Element is not found!");}
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int size = sc.nextInt();
        int[] arr = new int[size];
        System.out.println("Enter elements of the array (sorted): ");
        for (int i = 0; i < size; i++) arr[i] = sc.nextInt();
        System.out.print("Enter the key to find: ");
        int key = sc.nextInt();
        binarySearch(arr, key);}}
```

Output:



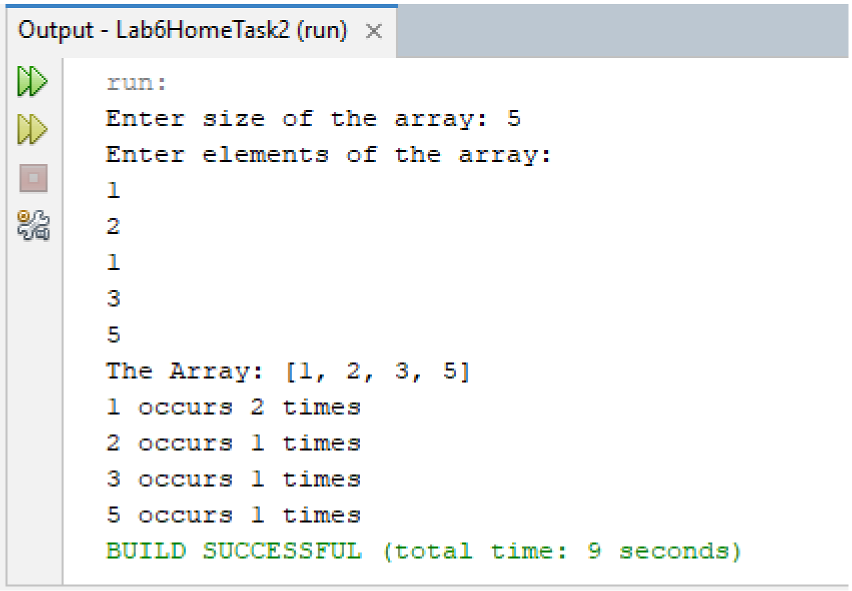
```
run:
Enter the size of the array: 3
Enter elements of the array (sorted):
95
65
6565
Enter the key to find: 65
Element is found at index: 1
BUILD SUCCESSFUL (total time: 16 seconds)
```

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”.

Source Code:

```
import java.util.*;
public class KhurramLab6HomeTask2 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter size of the array: ");
        int size = sc.nextInt();
        int[] arr = new int[size];
        System.out.println("Enter elements of the array: ");
        for (int i = 0; i < size; i++) arr[i] = sc.nextInt();
        Map<Integer, Integer> frequencyMap = new HashMap<>();
        for (int num : arr) {
            frequencyMap.put(num, frequencyMap.getOrDefault(num, 0) + 1);}
        System.out.println("The Array: " + frequencyMap.keySet());
        frequencyMap.forEach((key, count) -> System.out.println(key + " occurs " + count + " times"));}}
```

Output:



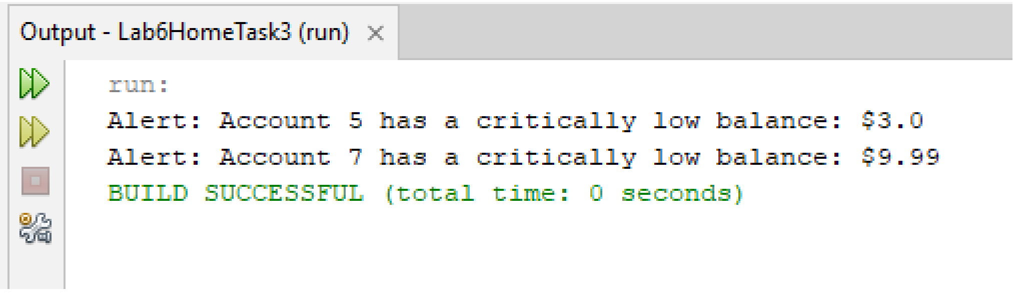
```
run:
Enter size of the array: 5
Enter elements of the array:
1
2
1
3
5
The Array: [1, 2, 3, 5]
1 occurs 2 times
2 occurs 1 times
3 occurs 1 times
5 occurs 1 times
BUILD SUCCESSFUL (total time: 9 seconds)
```

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.

Source Code:

```
public class KhurramLab6HomeTask3 {
    public static void checkLowBalances(double[] balances, double threshold) {
        for (int i = 0; i < balances.length; i++) {
            if (balances[i] < threshold) {
                System.out.println("Alert: Account " + (i + 1) + " has a critically low balance: $" + balances[i]);}}}
    public static void main(String[] args) {
        double[] balances = {1200.50, 50.75, 10.30, 500.00, 3.00, 15.00, 9.99};
        double threshold = 10.00;
        checkLowBalances(balances, threshold);}}
```

Output:



```
run:
Alert: Account 5 has a critically low balance: $3.0
Alert: Account 7 has a critically low balance: $9.99
BUILD SUCCESSFUL (total time: 0 seconds)
```