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

```java
package search;
public class Search {
    public static void main(String[] args) {
        int[] balances = new int[10];
        balances[0] = 5000;
        balances[1] = 25000;
        balances[2] = 15000;
        balances[3] = 9000;
        balances[4] = 1200000;
        balances[5] = 6000;
        balances[6] = 300000;
        balances[7] = 7000;
        balances[8] = 100000;
        balances[9] = 8000;
        for (int i = 0; i < 10; i++) {
            if (balances[i] < 10000) {
                System.out.println("Account No. " + (i + 1) + " Low Balance");
            }
        }
    }
}

```

Output:

```
run:
Account No. 1 Low Balance
Account No. 4 Low Balance
Account No. 6 Low Balance
Account No. 8 Low Balance
Account No. 10 Low Balance
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

```java
package search;
import java.util.Arrays;
public class task2 {
    public static void main(String[] args) {
        int[] arr= {12, 35, 7, 64, 23, 1, 89, 56};
        int tofind = 89;
        int result = Arrays.binarySearch(arr, tofind);
        if (result >= 0) {
            System.out.println("Element " + tofind + " found at index: " + result);
        } else {
            System.out.println("Element " + tofind+ " not found.");
        }
    }
}
```

Output:

```
run:
Element 89 found at index: 6
BUILD SUCCESSFUL (total time: 0 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.

```java
package search;
public class task3 {
    public static void bubbleSort(int[] arr) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - 1 - i; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }
    }
    public static int findMissingPositive(int[] arr) {
        bubbleSort(arr);
        int missing = 1;
        for (int i = 0; i < arr.length; i++) {
            if (arr[i] == missing) {
                missing++;
            }
        }
        return missing;
    }
    public static void main(String[] args) {
        int[] arr = {3, 4, 5, 1};

        System.out.println("The smallest missing positive integer is: " + findMissingPositive(arr));
    }
}
```

Output:

```
run:
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

```java
package sorting;
public class abiha {
    public static int findFirstOccurrence(int[] arr, int target) {
        int left = 0, right = arr.length - 1;
        int result = -1;
        while (left <= right) {
            int mid = (left + right) / 2;
            if (arr[mid] == target) {
                result = mid;
                right = mid - 1;
            } else if (arr[mid] > target) {
                right = mid - 1;
            } else {
                left = mid + 1;
            }
        }
        return result;
    }
    public static void main(String[] args) {
        int[] arr = {1, 2, 4, 4, 4, 5, 6, 7};
        int target = 4;

        int result = findFirstOccurrence(arr, target);
        System.out.println("First occurrence of " + target + " is at index: " + result);
    }
}
```

Output:

```
run:
First occurrence of 4 is at index: 2
BUILD SUCCESSFUL (total time: 0 seconds)
```

Home Task

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

```java
package sorting;
import java.util.Arrays;
public class hometaskab {
    public static void main(String[] args) {
        int[] arr = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39};
        int target = 15;
        int result = binarySearch(arr, target);
        if (result == -1) {
            System.out.println("Element " + target + " not found in the array.");
        } else {
            System.out.println("Element " + target + " found at index " + result + ".");
        }
    }
    public static int binarySearch(int[] arr, int target) {
        int low = 0;
        int high = arr.length - 1;
        while (low <= high) {
            int mid = (low + high) / 2;
            if (arr[mid] == target) {
                return mid;
            }
            if (arr[mid] < target) {
                low = mid + 1;
            }
            else {
                high = mid - 1;
            }
        }
        return -1;
    }
}
```

Output:

```
run:
Element 15 found at index 7.
BUILD SUCCESSFUL (total time: 0 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”.

```java
package sorting;
public class htask2ABIHA {
    public static void main(String[] args) {
        int[] Arr = {28, 1, 0, 1, 0, 3, 4, 0, 0, 3};
        occurrences(Arr);
    }
    public static void occurrences(int[] Arr) {
        for (int i = 0; i < Arr.length; i++) {
            boolean isDistinct = true;
            for (int j = 0; j < i; j++) {
                if (Arr[i] == Arr[j]) {
                    isDistinct = false;
                    break;
                }
            }
            if (isDistinct) {
                int count = 0;
                for (int k = 0; k < Arr.length; k++) {
                    if (Arr[k] == Arr[i]) {
                        count++;
                    }
                }
                System.out.print(Arr[i] + " " + count + "; ");
            }
        }
    }
}

```

Output:

```
run:
28 1; 1 2; 0 4; 3 2; 4 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.

```java
package sorting;
import java.util.Scanner;
public class bankacc {
    public static void main(String[] args) {
        Scanner abiha = new Scanner(System.in);
        System.out.print("Enter the number of accounts: ");
        int numAcc =abiha.nextInt();
        double[] accbal = new double[numAcc];
        System.out.println("Enter the balances for each account:");
        for (int i = 0; i < numAcc; i++) {
            System.out.print("Balance for account " + (i + 1) + ": ");
            accbal[i] = abiha.nextDouble();
        }
        System.out.print("Enter the threshold for critically low balance: ");
        double threshold = abiha.nextDouble();
        identify(accbal, threshold);
    }
    public static void identify(double[] balance, double threshold) {
        for (int i = 0; i < balance.length; i++) {
            if (balance[i] < threshold) {
                System.out.println("Account " + (i + 1) + " has a critically low balance of $" + balance[i]);
            }
        }
    }
}
```

Output:

```
run:
Enter the number of accounts: 5
Enter the balances for each account:
Balance for account 1: 45678
Balance for account 2: 098544
Balance for account 3: 234561
Balance for account 4: 65410976
Balance for account 5: 120097
Enter the threshold for critically low balance: 50000
Account 1 has a critically low balance of $45678.0
BUILD SUCCESSFUL (total time: 41 seconds)
```