# LAB # 06

**LAB TASKS 1:**

public class Main{

public static void main(String[] args) {

// Declare and initialize an array of size 10

int[] accountBalances = new int[10];

for (int i = 0; i < accountBalances.length; i++) {

accountBalances[i] = (int)(Math.random() \* 100000); // Generates a random number between 0 and 1,000,000

}

for (int i = 0; i < accountBalances.length; i++) {

if (accountBalances[i] < 1000000) {

System.out.println("Account no: " + (i + 1) + " has Low Balance: " + accountBalances[i]);

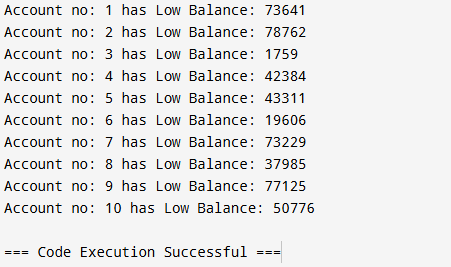
}

}

}

}

**OUTPUT**



**LAB TASKS 2:**

import java.util.Arrays;

public class Main {

public static void main(String[] args) {

// Define an array of integers

int[] numbers = {20, 10, 30, 50, 40};

// Number to search for

int target = 30;

// Sort the array before using binarySearch (required for binarySearch)

Arrays.sort(numbers);

// Search for the target using Arrays.binarySearch

int index = Arrays.binarySearch(numbers, target);

// Check the result

if (index >= 0) {

System.out.println("Number " + target + " found at index " + index);

} else {

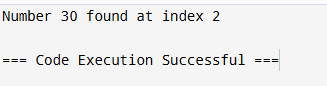
System.out.println("Number " + target + " not found in the array.");

}

}

}

**OUTPUT**



**LAB TASKS 3:**

import java.util.Arrays;

public class Main {

public static void main(String[] args) {

int[] arr = {-1, 3, 5, 2, -2, 1, 4};

int Mussab = findSmallestMissingPositive(arr);

System.out.println("The smallest missing positive integer is: " + Mussab);

}

public static int findSmallestMissingPositive(int[] arr) {

Arrays.sort(arr);

int missing = 1;

for (int num : arr) {

if (num == missing) {

missing++;

}

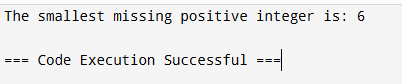
}

return missing;

}

}

**OUTPUT**



**LAB TASKS 4:**

public class Main {

// Method to find the first occurrence of the target element

public static int findFirstOccurrence(int[] arr, int target) {

int low = 0;

int high = arr.length - 1;

while (low <= high) {

int mid = low + (high - low) / 2;

if (arr[mid] == target) {

// Check if it is the first occurrence

if (mid == 0 || arr[mid - 1] != target) {

return mid; // Found the first occurrence

} else {

high = mid - 1; // Continue searching in the left half

}

} else if (arr[mid] < target) {

low = mid + 1; // Target is in the right half

} else {

high = mid - 1; // Target is in the left half

}

}

return -1; // Target not found

}

// Main method to test the function

public static void main(String[] args) {

int[] arr = {1, 2, 2, 2, 3, 4, 5};

int target = 2;

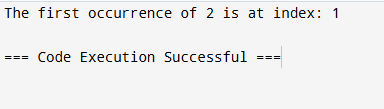
int result = findFirstOccurrence(arr, target);

System.out.println("The first occurrence of " + target + " is at index: " + result);

}

}

**OUTPUT**



**HOME TASK 1:**

import java.util.Arrays;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

// Initialize an array of size 20

int[] arr = new int[20];

// Populate the array with random values (sorted for binary search)

for (int i = 0; i < arr.length; i++) {

arr[i] = i \* 2; // Filling with multiples of 2 for simplicity

}

// Display the array

System.out.println("Array: " + Arrays.toString(arr));

// Get the target element to search for

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the element to search: ");

int target = scanner.nextInt();

// Perform binary search

int result = binarySearch(arr, target);

// Output the result

if (result != -1) {

System.out.println("Element " + target + " found at index: " + result);

} else {

System.out.println("Element " + target + " not found in the array.");

}

scanner.close();

}

// Binary Search Method

public static int binarySearch(int[] arr, int target) {

int low = 0;

int high = arr.length - 1;

while (low <= high) {

int mid = low + (high - low) / 2;

if (arr[mid] == target) {

return mid; // Target found

} else if (arr[mid] < target) {

low = mid + 1; // Narrow search to the right half

} else {

high = mid - 1; // Narrow search to the left half

}

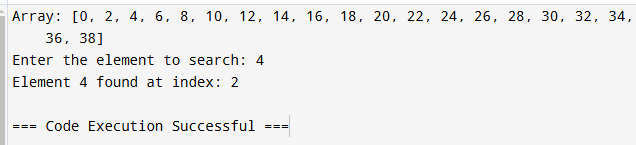
}

return -1; // Target not found

}

}

**OUTPUT**



**HOME TASK 2:**

import java.util.HashMap;

import java.util.Map;

public class OccurrencesExample {

public static void main(String[] args) {

// Example array

int[] A = {28, 1, 0, 1, 0, 3, 4, 0, 0, 3};

// Call the occurrences function

occurrences(A);

}

// Function to count and print occurrences of each distinct value

public static void occurrences(int[] A) {

// Use a HashMap to store the frequency of each number

Map<Integer, Integer> frequencyMap = new HashMap<>();

// Populate the frequency map

for (int num : A) {

frequencyMap.put(num, frequencyMap.getOrDefault(num, 0) + 1);

}

// Print each distinct value and its number of occurrences

for (Map.Entry<Integer, Integer> entry : frequencyMap.entrySet()) {

System.out.print(entry.getKey() + " " + entry.getValue() + "; ");

}

}

}

**OUTPUT**



**HOME TASK 3:**

import java.util.ArrayList;

import java.util.List;

public class Main {

public static void main(String[] args) {

// Example account balances

double[] balances = {1200.0, 60.5, 200.0, 20.0, 1800.0, 5.0};

double threshold = 100.0;

// Call the function to identify low-balance accounts

List<Integer> lowBalanceAccounts = identifyLowBalances(balances, threshold);

// Output the results

System.out.println("Accounts with critically low balances (below $" + threshold + "):");

for (int accountIndex : lowBalanceAccounts) {

System.out.println("Account #" + accountIndex + " has a balance of $" + balances[accountIndex]);

}

}

// Function to identify accounts with balances below a threshold

public static List<Integer> identifyLowBalances(double[] balances, double threshold) {

List<Integer> lowBalanceIndices = new ArrayList<>();

for (int i = 0; i < balances.length; i++) {

if (balances[i] < threshold) {

lowBalanceIndices.add(i);

}

}

return lowBalanceIndices;

}

}

**OUTPUT**

